Numerical results of quasi-newton methods for unconstrained function minimization

Heuts, R.M.J.; Vandaele, W.H.

Publication date:
1971

R. M. J. Heuts and W. H. Vandaele

Numerical results of quasi-newton methods for unconstrained function minimization

Research memorandum

TILBURG INSTITUTE OF ECONOMICS
DEPARTMENT OF ECONOMETRICS

25 juni, 1971.

NUMERICAL RESULTS OF QUASI-NEWTON METHODS FOR UNCONSTRAINED FUNCTION MINIMIZATION

by R.M.J.HEUTS and W.H.VANDAELE *

The authors are grateful to Dr.M.Powell for some comments on the step-length procedure. They also thank Dr.W.Molenaar and Mr.D.Neeleman for some general comments on the draft.

* At present Ford Foundation European Doctoral Fellow and Honorary Graduate Fellow of the Belgian American Education Foundation (C.R.B.) at the University of Chicago, Chicago, Illinois 60637.

SUMMARY

For unconstrained function minimization we have tested two interpolation procedures, which are used to calculate the steplength along the direction vector. The two procedures are: cubic interpolation and quadratic interpolation. Five testfunctions and several initial estimates are used to see which procedure is the best.

1. Introduction

The method discussed here, is based on an algorithm of Davidon W.C. [6] and extended by Fletcher R. and M.J.D.Powell [7], Barnes J.G.P. - Rosen E.M. [1] [18], and later generalised in a series of articles by Shanno D.F. [19], and Shanno D.F. and P.C. Kettler [20]. It is an iterative procedure to find the relative minimum of a function of several variables without constraints.

We are considering a standard quadratic form in p-dimensions

$$Q(x;\ \theta) = Q_o + \sum_{i=1}^{p} \theta_i\ x_i + \tfrac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} S_{ij}\ \theta_i \theta_j, \ \ldots \qquad (1.1)$$

which is supposed to have a minimum, for which Fletcher R. and M.J.D.Powell proved that it will be reached by the algorithm in p iterations.
The so called variable metric algorithm was developed by Davidon W.C.[6]. It is now one of the most frequently used and most sucessful techniques for calculating the minimum of a differentiable function of several variables.
Powell M.J.D. [16] has proved convergence in the case that the objectfunction has continuous second derivatives, and satisfies a strict convexity condition.
Previously convergence had been proved only in the very special case when the objectfunction is exactly quadratic, which is not always the case.
Kowalik J. and M.R. Osborne [13] state that the algorithm due to Davidon would seem to be the best general purpose optimization procedure making use of derivatives that is currently available.
Box M.J. [2] in a paper about comparison of several current unconstrained optimization methods comes to the conclusion that the method due to Davidon was the most consistently successful.
In this paper we have compaired a quadratic and cubic interpolation procedure for the steplength. Several methods of scaling of the matrix $H^{(k)}$ (see section 2.4) and 5 testfunctions are used to come to a conclusion about the two interpolation methods.

2.1 Method of computation

The function $Q(x; \theta)$ is minimized as follows:

- Given the parameter vector $\theta^{(k)}$ we compute the gradient vector

$$g^{(k)} \equiv \left(\frac{\partial Q(x; \theta)}{\partial \theta}\right)_{\theta = \theta^{(k)}} \quad , \text{ a } p \times 1 \text{ matrix,}$$

where k is the k-th stage of the algorithm.

- Thereafter we determine the direction vector

$$s^{(k)} \equiv s^{(k)}(x; \theta) = -H^{(k)} g^{(k)}$$

The determination of the matrix $H^{(k)}$ will be discussed below.

- Next we search along $-H^{(k)}g^{(k)}$ to find a scalar $\alpha_{(k)}$ such that $Q(x; \theta^{(k)} + \alpha_{(k)} s^{(k)})$ is a minimum. Fletcher R. and M.J.D.Powell have proved that $\alpha_{(k)}$ may be a positive scalar [7].

- The scalar $(\alpha_{(k)})_{min}$ is used to determine the steplength along the line $s^{(k)}$:

$\sigma^{(k)} = (\alpha_{(k)})_{min} s^{(k)}$, after which we can compute the new vector of parameters

$$\theta^{(k+1)} = \theta^{(k)} + \sigma^{(k)}.$$

- Set $z^{(k)} = g^{(k+1)} - g^{(k)}$

- Compute

$$H^{(k+1)} = H^{(k)} + t\,\frac{\sigma^{(k)}\sigma^{(k)'}}{\sigma^{(k)'}z^{(k)}} + \frac{[(1-t)\sigma^{(k)} - H^{(k)}z^{(k)}][(1-t)\sigma^{(k)} - H^{(k)}z^{(k)}]'}{[(1-t)\sigma^{(k)} - H^{(k)}z^{(k)}]'z^{(k)}}$$

The use of the parameter t will be discussed later.

- The algorithm is repeated until a minimum is reached.

In the first step of the iteration it is customary to set $H^{(o)} = I$

so that the first step is equivalent to a step of the steepest descent method.
If $H^{(o)}$ is positive definite, then it can be proved that all subsequent $H^{(k)}$ are also positive definite [19].

2.2. Starting value for the parameter vector $\theta^{(o)}$

For the computation of starting values $\theta^{(o)}$ we can take a number of arbitrary parameter vectors in the p-dimensional parameter space and compute for every vector in this space the function value of $Q(x; \theta)$.
That vector for which the object function is a minimal, will be taken as a starting vector for the iterative procedure.

2.3 The determination of $\alpha_{(k)}$

The scalar parameter $\alpha_{(k)}$ can be found by fitting a second or third degree polynomial in $\alpha_{(k)}$ to the Q-values and to determine the value of $\alpha_{(k)}$ for which this polynomial is a minimum. Details of the quadratic and cubic interpolation procedure can be found in appendix I and II.

2.4. The determination of the matrix $H^{(k)}$

Newton's method for minimizing a function $Q(x; \theta)$, where θ is a p-vector, is to generate a sequence of points

$$\theta^{(k+1)} = \theta^{(k)} - \alpha_{(k)} [S^{(k)}]^{-1} g^{(k)}, \ldots\ldots \qquad (2.4.1)$$

where $g^{(k)} = \left(\frac{\partial Q(x; \theta)}{\partial \theta}\right)_{\theta = \theta^{(k)}}$, $S^{(k)} = \left(\frac{\partial^2 Q(x; \theta)}{\partial \theta_i \partial \theta_j}\right)$,

the Hessian matrix of $Q(x; \theta)$ evaluated at $\theta^{(k)}$, and $\alpha_{(k)}$ an appropriately chosen scalar.
Quasi-Newton methods use an initial approximation and generate an approximation $H^{(k)}$ to $[S^{(k)}]^{-1}$ at each step rather than performing the computational work of evaluating and inverting $S^{(k)}$, such as by Hartley's method [11].

The sequence (2.4.1) then becomes

$$\theta^{(k+1)} = \theta^{(k)} - \alpha_{(k)} H^{(k)} g^{(k)}. \qquad \ldots\ldots\ldots(2.4.2)$$

The determination of $\alpha_{(k)}$ is discussed in 2.3.
Some well-known techniques of this type are the Fletcher-Powell modification of Davidon's method [7] , Broyden methods [3] ,[4] , the Barnes-Rosen method [1] ,[18] and Goldfarb's method [9] .
The Fletcher-Powell technique guarantees that the matrix $H^{(k)}$ will always be positive definite.
Shanno's method [19] develops a family of matrices $H^{(k)}$ as a function of a scalar parameter t, and it can be shown that both the Fletcher-Powell and Barnes-Rosen matrices are special cases of this parametric family.
The technique for generating a series of approximations $H^{(k)}$ to the inverse of the Hessian at the points $\theta^{(k)}$ can be described as follows:
Assume $Q(x; \theta)$ is a positive definite quadratic form, $H^{(k)}$ the current approximation to the inverse of the Hessian, and $S^{(k)}$ the approximation to the Hessian.
Assume $z^{(k)} = g^{(k+1)} - g^{(k)}$. If $S^{(k)}$ is a constant matrix, we have

$g^{(k)} = \nabla Q = x + S\,\theta^{(k)}$ (see 1.1), and

$$z^{(k)} = S(\theta^{(k+1)} - \theta^{(k)}) = S\,\sigma^{(k)} . \qquad \ldots\ldots\ldots(2.4.3)$$

Multiplying (2.4.3) by $H^{(k)}$, we obtain

$$H^{(k)} z^{(k)} = \sigma^{(k)}. \qquad \ldots\ldots\ldots(2.4.4)$$

Since (2.4.4) in general will not be satisfied, assume the error lies in $H^{(k)}$. The matrix $H^{(k)}$ has to a certain extent the properties of the inverse of the S-matrix. We then modify $H^{(k)}$ by introducing a matrix of additive corrections $D^{(k)}$ such that

$$(H^{(k)} + D^{(k)})\, z^{(k)} = \sigma^{(k)} \qquad \ldots\ldots\ldots(2.4.5)$$

or

$$D^{(k)} z^{(k)} = \sigma^{(k)} - H^{(k)} z^{(k)} \qquad \ldots\ldots\ldots(2.4.6)$$

To restrict the choice of t, Shanno [19] has proved the following:

If $H^{(k)}$ is positive definite, $\forall\ t > \frac{\alpha_{(k)} - 1}{\alpha_{(k)}}$, also $H^{(k+1)}$ is positive definite.

For $t > \frac{\alpha_{(k)} - 1}{\alpha_{(k)}}$, $H^{(k+1)}$ is positive definite, hence at no finite step does the smallest eigenvalue of $H^{(k+1)}$ ever become zero. However, it is possible that if λ_1 is the smallest eigenvalue of $H^{(k+1)}$, $\lim_{k\to\infty} \lambda_1 = 0$.

This can only be due to computer rounding errors. In this case the iterative technique will degenerate as $k \to \infty$.

To attempt to alleviate this difficulty, we may at each step choose t in such a way as to maximize the smallest eigenvalue of $H^{(k+1)}$.

This is accomplished be choosing t to maximize $v'H^{(k+1)}v$ for any arbitrary vector v.

We shall show this as follows:

Say the matrix $H^{(k+1)}$ is positive definite, then maximizing the quadratic form $v'H^{k+1}v$ means maximizing $\sum_j \lambda_j v_j^2$, where the λ_j are positive charateristic roots of the matrix $H^{(k+1)}$. Now we can say that maximizing $\sum_j \lambda_j v_j^2$ means also maximizing λ_1, the smallest charateristic root of the matrix $H^{(k+1)}$.

Shanno [19] has proved the following interesting theorem:

Let v be an arbitrary vector. Then $v'H^{(k+1)}v$ is a non-decreasing function of t.

It can be shown that the condition of $H^{(k+1)}$ improves monotonically with t. This necessitates finding a closed form representation of $H^{(k+1)}$ for $t = \infty$.

It can be proved that this representation is as follows [19]:

Let $H^{(k+1)}$ be defined by (2.4.10), and let

$$r = \frac{\sigma^{(k)'} z^{(k)}}{\sigma^{(k)'} z^{(k)} + z^{(k)'} H^{(k)} z^{(k)}}$$

Then

$$\lim_{t\to\infty} H^{(k+1)} = H^{(k)} + \frac{(\sigma^{(k)} - r\, H^{(k)} z^{(k)})(\sigma^{(k)} - r\, H^{(k)} z^{(k)})'}{(\sigma^{(k)} - r\, H^{(k)} z^{(k)})'\, z^{(k)}} +$$

If we let $H^{(k+1)} = H^{(k)} + D^{(k)}$ and $D^{(k)}$ is chosen to satisfy (2.4.6), we then have

$$H^{(k)}\, z^{(k)} = \sigma^{(k)} \qquad \text{.....(2.4.7)}$$

Fletcher and Powell [7] have proved that when $Q(x; \theta)$ is a positive definite quadratic form, and at each step (2.4.7) is satisfied, then the minimum of $Q(x; \theta)$ will be reached in at most p iterations.
For functions which are not quadratic, but strict convex, the convergence can also be proved [16].
To determine $D^{(k)}$ we can write (2.4.6) in the form

$$D^{(k)}\, z^{(k)} = t\, \sigma^{k} + (1-t)\, \sigma^{(k)} - H^{(k)}\, z^{k} \qquad \text{.....(2.4.8)}$$

which by multiplying with the scalars

$$\sigma^{(k)'} z^{(k)} \text{ and } ((1-t)\, \sigma^{(k)} - H^{(k)} z^{(k)})'\, z^{(k)}$$

becomes

$$D^{(k)} z^{(k)} = t\, \frac{\sigma^{(k)} \sigma^{(k)'} z^{(k)}}{\sigma^{(k)'} z^{(k)}} + \frac{[(1-t)\, \sigma^{(k)} - H^{(k)} z^{(k)}][(1-t)\, \sigma^{(k)} - H^{(k)} z^{(k)}]'}{[(1-t)\, \sigma^{(k)} - H^{(k)} z^{(k)}]'\, z^{(k)}}$$

So

$$D^{(k)} = t\, \frac{\sigma^{(k)} \sigma^{(k)'}}{\sigma^{(k)}\, z^{(k)}} + \frac{((1-t)\sigma^{(k)} - H^{(k)} z^{(k)})((1-t)\sigma^{(k)} - H^{(k)} z^{(k)})'}{((1-t)\sigma^{(k)} - H^{(k)} z^{(k)})'\, z^{(k)}}$$

$$\text{.....(2.4.9)}$$

Then $t = 0$ is the Barnes-Rosen choice, and $t = 1$ the Fletcher-Powell choice.
So we have

$$H^{(k+1)} = H^{(k)} + t\, \frac{\sigma^{(k)} \sigma^{(k)'}}{\sigma^{(k)'} z^{(k)}} +$$

$$\frac{[(1-t)\sigma^{(k)} - H^{(k)} z^{(k)}][(1-t)\sigma^{(k)} - H^{(k)} z^{(k)}]'}{[(1-t)\sigma^{(k)} - H^{k} z^{(k)}]'\, z^{(k)}} + \text{.....(2.4.10)}$$

$$+ (r-1) \frac{H^{(k)} z^{(k)} z^{(k)'} H^{(k)}}{z^{(k)'} H^{(k)} z^{(k)}} .$$

Powell [16] has proved under general conditions that in the variable metric algorithm

$$\| \theta^{(k+1)} - \theta^{*} \| \leq M \| \theta^{(k)} - \theta^{*} \| ,$$

where θ^{*} is the value of θ minimizing $Q(x;\theta)$ and $\| \ \|$ is the Euclidean vector norm.

This suggests using $\|\sigma^{(k)}\| = \|\alpha_{(k)} s^{(k)}\| = \|-\alpha_{(k)} H^{(k)} g^{(k)}\|$ as an estimate of $\|\theta^{(k)} - \theta^{*}\|$ and t is chosen so that

$\|s^{(k+1)}\| = \|\sigma^{(k)}\|$. This version is called the constant norm version.

3. Conclusions

The quadratic interpolation procedure uses many function evaluations, but sometimes this is useful especially for the Weibull function, because for this function the quadratic procedure always converged, except for one case; but the cubic interpolation often failed or many iterations had to be used for this testfunction. To our opinion it is perhaps the best to use the cubic interpolation for simple functions, but for particularly badly conditioned problems our experience is to use the quadratic procedure.

To Mr. M.J.D. Powell's [17] opinion the most recent work on methods for unconstrained optimization is that which explores algorithms that avoid the subproblem of minimizing a function of one variable on every iteration. He refers to an algorithm of M.R. Hestenes [12] that does not make any linear searches.

Appendix I: Quadratic interpolation for minimizing a function of a single variable

A simple assumption about the behaviour of a nonlinear function is that in the neighborhoud of its minimum, the function can adequately be represented by a quadratic polynomial.
Note the close relation with the assumption made in the introduction about the function to optimize. There it was also stated that in the neighborhoud of its minimum this function can be approximated by a quadratic form.
The procedure for minimizing a function using a quadratic interpolation is as follows: evaluate the function at three points, fit a quadratic interpolation polynomial to it, and calculate the minimum of this interpolant. This minimum replaces one of the initial points and the procedure is repeated using this new function value until a suitable convergence criterion is satisfied.
(see figure 1).
More specific we make use of the Gregory-Newton theorem (see [21, p.48]):
Given n + 1 equidistant* points, it is possible to fit a polynomial of degree n or less, $P_n(x)$, which goes through the n + 1 points:

$$P_n(x) = y_1 + \phi_1 \Delta y_1 + \phi_2 \frac{\Delta^2 y_1}{2!} + \ldots\ldots + \phi_n \frac{\Delta^n y_n}{n!} \qquad \ldots\ldots (I\text{-}1)$$

where:

$$\phi_n = \frac{(x - x_1)(x - x_2)\ldots(x - x_n)}{d^n} ;$$

and d is the distance between the equidistant points.
Suppose we fit a second degree polynomial to three points. Then the polynomial has the following form

$$P_2(x) = y_1 + \phi_1 \Delta y_1 + \phi_2 \frac{\Delta^2 y_1}{2!} \qquad \ldots\ldots (I\text{-}2)$$

---

* It is also possible to fit a polynomial if the points are not equidistant.
However introducing equidistant points simplifies the procedure considerable.

where

$$\Delta y_1 = y_2 - y_1 \qquad \text{with } y_i = f(x_i) : \text{value of the function at } x_i$$

$$\Delta^2 y_1 = y_3 - 2y_1 + y_1$$

$$\phi_1 = \frac{x - x_1}{d} \qquad \ldots\ldots \text{(I-3)}$$

$$\phi_2 = \frac{(x - x_1)(x - x_2)}{d^2}$$

or

$$P_2(x) = y_1 + \frac{(x - x_1)}{d} (y_2 - y_1) + \frac{(x - x_1)(x - x_2)}{d^2} \frac{(y_3 - 2y_2 + y_1)}{2!} \qquad \text{(I-4)}$$

A minimum is reached for

$$x_{min} = \tfrac{1}{2} (x_1 - x_2) - \frac{d\,(y_2 - y_1)}{(y_3 - 2y_2 + y_1)} \qquad \ldots\ldots \text{(I-5)}$$

if the function satisfies the convexity criterion:

$y_3 + y_1 > 2y_2$.

Transformed to the original variables (see above: Method of computation) we find for the optimum of the steplength:

$$\alpha^* = (\alpha_{(k)})_{min} = \tfrac{1}{2} (\alpha^I_{(k)} + \alpha^{II}_{(k)}) - \frac{d\,\{Q(\alpha^{II}) - Q(\alpha^I)\}}{Q(\alpha^{III}) - 2Q(\alpha^{II}) + Q(\alpha^I)} \qquad \ldots \text{(I-6)}$$

provided that $Q(\alpha^{III}) + Q(\alpha^I) > 2\,Q(\alpha^{II})$ $\qquad \ldots\ldots \text{(I-7)}$

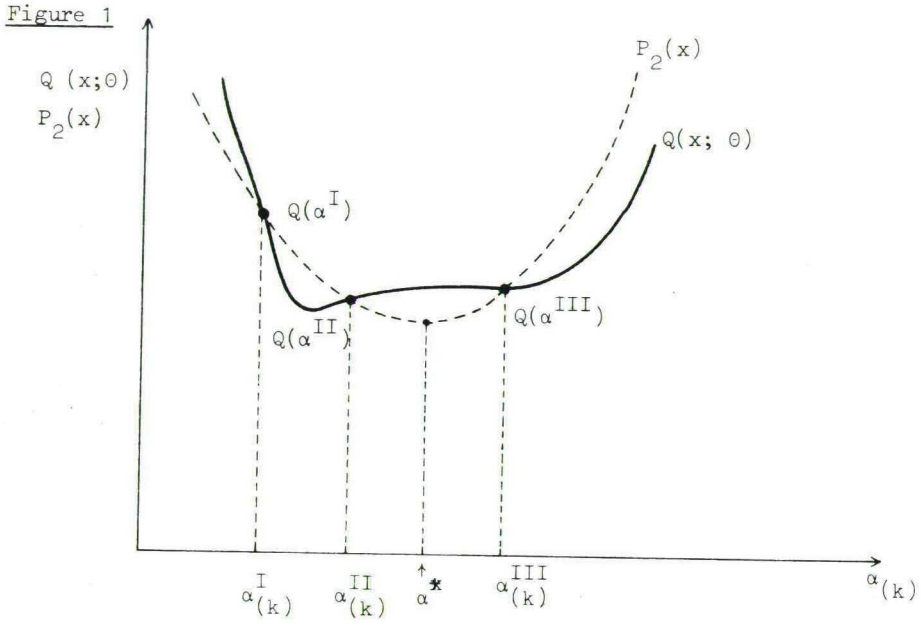
where:

$\alpha^I_{(k)}$, $\alpha^{II}_{(k)}$, $\alpha^{III}_{(k)}$: three equidistant points;

d the fixed distance between two such points;

$Q(\alpha^I) \equiv Q(x;\ \theta^{(k)} + \alpha^I_{(k)}\, s^{(k)})$.

Figure 1



$\alpha^* \equiv (\alpha_{(k)})_{min}$ is then substituted back in the function that we are minimizing, that is $Q(\alpha^*)$ is calculated and its derivative $Q'(\alpha^*)$. According as Q' is positive or negative, the interpolation is repeated over one of the subintervals:

$(\alpha^I_{(k)}, \alpha^*)$ or $(\alpha^*, \alpha^{III}_{(k)})$ respectively.

This is illustrated in the following two figures.

Figure 2

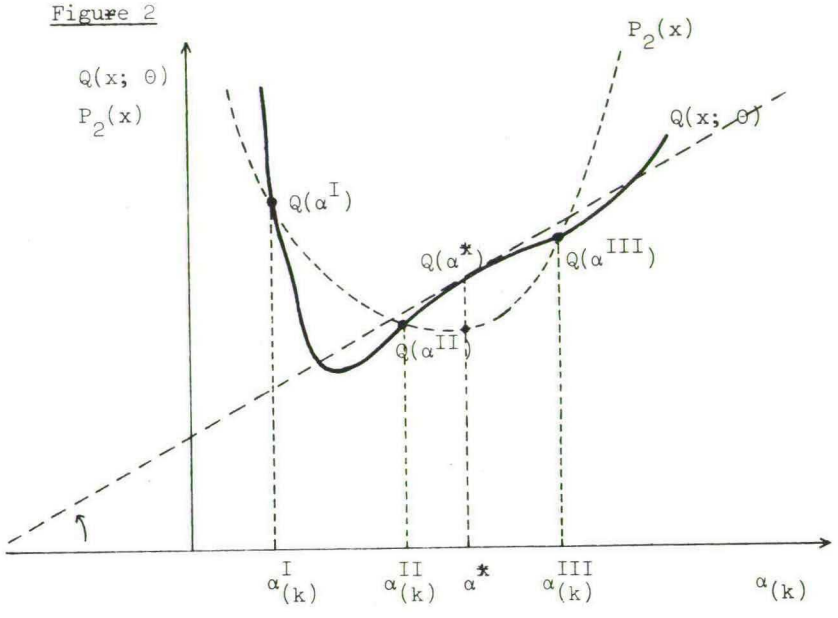
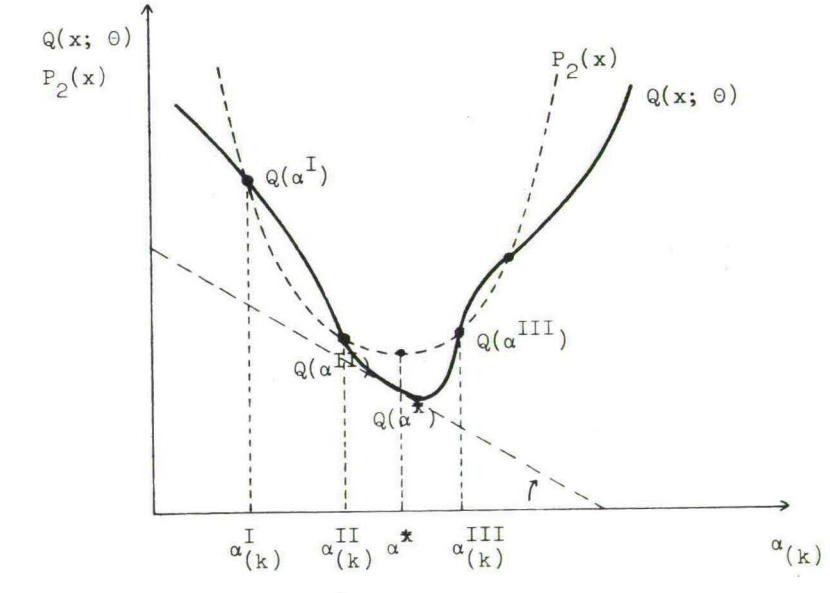


Figure 3



Remarks:

It can be possible that the convexity criterion is satisfied but that we go on increasing or decreasing the steplength before we fit the polynomial. Increasing is done when the derivative of the function in $\alpha^{III}_{(k)}$, $Q'(\alpha^{III})$, is negative; and decreasing is done when $Q(\alpha^{III})$ and $Q(\alpha^{II})$ are greater than $Q(\alpha^{I})$.

Appendix II: Davidon's cubic interpolation for minimizing a function of a single variable [6]

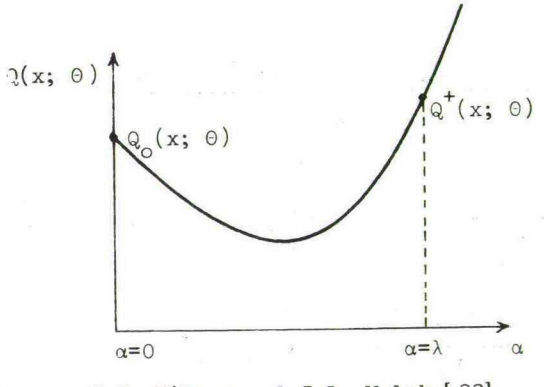
Inasmuch as the interpolation is along a one-dimensional interval, it is convenient to plot the function along this direction as a simple graph (see figure 4).
The values of $Q_o(x; \theta)$ and $Q^+(x; \theta)$ of the function at points α_{j-1} and α_j are known, and so are its slopes g_k and g_k^+, at these two points. The interpolation for the location of the minimum is based on spline-function technique*: choosing the smoothest curve satisfying the boundary conditions at α_{j-1} and α_j, where the smoothest curve is defined as the curve which minimizes

$$\int_o^\lambda \left(\frac{d^2 Q(x; \theta)}{d \alpha^2}\right)^2 d\alpha \qquad \ldots\ldots(II-1)$$

where (II-1) is a norm defined for spline-functions.
It can be proved that this norm is minimized when a cubic is fitted.
Third degree spline functions have been used quite extensively since the 1959 Davidon paper [6], have generally given satisfactory results, and are easy to calculate.

Derivation

Consider the following figure
Figure 4



* See J.H. Ahlberg, E.N. Nilson and J.L. Walsh [23].

Suppose the third degree spline-function is of the following form

$$S_p(\alpha) = \frac{1}{3} a \alpha^3 + \frac{1}{2} b \alpha^2 + c \alpha + d \qquad \ldots\ldots (II-2)$$

and

$$S_p'(\alpha) = a \alpha^2 + b \alpha + c$$

$$S_p''(\alpha) = 2 a \alpha + b \qquad \ldots\ldots (II-3)$$

Define

$$S_p'(0) = g_k \qquad S_p(0) = Q_o(x; \theta)$$

$$S_p'(\lambda) = g_k^+ \qquad S_p(\lambda) = Q^+(x; \theta) \qquad \ldots\ldots (II-4)$$

With these four points it is possible to determine the parameters a, b, c, d, using spline-function theory, and the function $S_p(\alpha)$ is now expressed as

$$S_p(\alpha) = g_k \frac{(\lambda-\alpha)^2 \alpha}{\lambda^2} - g_k^+ \frac{\alpha^2 (\lambda-\alpha)}{\lambda^2} + Q_o(x; \theta) \frac{(\lambda-\alpha)^2 (2\alpha+\lambda)}{\lambda^3}$$

$$+ Q^+(x; \theta) \frac{\alpha^2 [2(\lambda-\alpha) + \lambda]}{\lambda^3} \qquad \ldots\ldots (II-5)$$

and

$$S_p'(\alpha) = \alpha^2 \left\{ \frac{3 g_k \lambda + 3 g_k^+ \lambda + 6 Q_o(x; \theta) - 6 Q^+(x; \theta)}{\lambda^3} \right\} +$$

$$+ \alpha \left\{ \frac{- 4 g_k \lambda - 2 g_k^+ \lambda - 6 Q_o(x; \theta) + 6 Q^+(x; \theta)}{\lambda^2} \right\} + g_k \ldots\ldots (II-6)$$

or

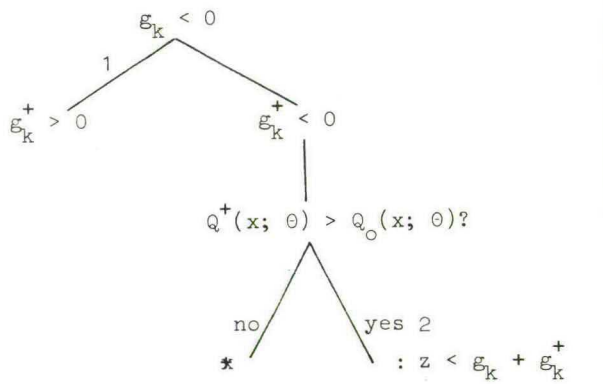
$$S_p'(\alpha) = g_k - \frac{2\alpha}{\lambda} (g_k + z) + \frac{\alpha^2}{\lambda^2} (g_k + g_k^+ + 2z) \qquad \ldots\ldots (II-7)$$

where

$$z = \frac{3 [Q_o(x; \theta) - Q^+(x; \theta)]}{\lambda} + g_k + g_k^+ .$$

The root of eq. (II-7) that corresponds to a minimum lies between 0 and λ in virtue of the fact that $g_k < 0$ and either $g_k^+ > 0$ or $z < g_k + g_k^+$. We can make the following diagram

diagram 1



* In this case we will not interpolate: as $Q^+(x; \theta) < Q_o(x; \theta)$ and $g_k^+ < 0$, we can still reduce the function value simply by increasing the steplength.

The results given by Davidon [6, p.10] are:

$$\alpha_{min} = \lambda (1-a) \qquad .....(II-8)$$

where

$$a = \frac{g_k^+ + P - z}{g_k^+ - g_k + 2P} \qquad .....(II-9)$$

and

$$P = (z^2 - g_k g_k^+)^{\frac{1}{2}} \qquad .....(II-10)$$

In order to verify the result (II-8), we will distinguish between a linear and a quadratic function (II-7).

a. Linear case

In the linear case formula (II-7) is:

$$S_p'(\alpha) = g_k - \frac{2\alpha}{\lambda}(g_k + z) \qquad \text{.....(II-11)}$$

So we have $g_k + g_k^+ + 2z = 0$, so that

$$z = -\frac{g_k + g_k^+}{2} \qquad \text{.....(II-12)}$$

Furthermore, equating $S_p'(\alpha) = 0$, we have for the optimal value of the steplength:

$$\alpha_{min} = \lambda \frac{g_k}{2(g_k+z)} = \lambda \frac{g_k}{2\left[g_k - \frac{g_k + g_k^+}{2}\right]} = \lambda \frac{g_k}{g_k - g_k^+} = \lambda\ (1\text{-}a)$$

.....(II-13)

where

$$a = \frac{g_k^+}{g_k^+ - g_k} \,. \qquad \text{.....(II-14)}$$

Before we verify the Davidon's results (II-8) we note that because of (II-12), it is impossible to be in branch 2 of diagram 1, and this can be seen as follows. In branch 2, $z < g_k + g_k^+$, g_k and g_k^+ being negative we also have

$$z < \frac{g_k + g_k^+}{2}$$

and it remains true, if we make the righthand side positive, that

$$z < -\frac{g_k + g_k^+}{2} \,. \qquad \text{.....(II-15)}$$

But this conflicts with (II-12). (Q.E.D.)

In order to check Davidon's result we have to prove that (II-14) equals (II-9). We first evaluate (II-10)

$$P = \sqrt{z^2 - g_k\, g_k^+} = \sqrt{\frac{g_k^2 + 2g_k\, g_k^+ + g_k^+ - 4g_k\, g_k^+}{4}} =$$

$$= \left| \frac{g_k - g_k^+}{2} \right| ,$$

and because we have just shown that we are in branch 1 of diagram 1, so with $g_k < 0$, $g_k^+ > 0$, $-\frac{g_k - g_k^+}{2}$ is positive and substituting this value in (II-9) together with (II-12) we get

$$a = \frac{g_k^+}{g_k^+ - g_k} , \qquad \text{.....(II-16)}$$

so that the Davidon method is correct in the linear case.

b. Quadratic case

Formula (II-7):

$$S_p'(\alpha) = g_k - \frac{2\alpha}{\lambda}(g_k + z) + \frac{\alpha^2}{\lambda^2}(g_k + g_k^+ + 2z) \qquad \text{.....(II-7)}$$

can be written as

$$\left.\begin{aligned} S_p'(\alpha) &= C + B\frac{\alpha}{\lambda} + A\frac{\alpha^2}{\lambda^2} \\ \text{where } C &= g_k \\ B &= -2(g_k + z) \\ A &= g_k + g_k^+ + 2z \end{aligned}\right\} \qquad \text{.....(II-17)}$$

The discriminant of equation (II-17) is

$$D = B^2 - 4AC. \qquad \text{.....(II-18)}$$

We shall first show that in branch 1 and 2 the discriminant is positive.

branch 1:

$$D = (-2\, g_k - 2z)^2 - 4(g_k + g_k^+ + 2z)g_k =$$

$$4\, z^2 - 4\, g_k\, g_k^+ > 0$$

branch 2:

$$D = 4z^2 - 4\, g_k\, g_k^+ > 4\, (g_k + g_k^+)^2 - 4\, g_k\, g_k^+ =$$

$$4\, g_k^2 + 4(g_k^+)^2 + 8\, g_k\, g_k^+ - 4\, g_k\, g_k^+ =$$

$$4\, g_k^2 + 4\, (g_k^+)^2 + 4\, g_k\, g_k^+ > 0.$$

The solutions of the quadratic form (II-17) are

$$\left(\frac{\alpha}{\lambda}\right)_{1,2} = \frac{-\, B \pm \sqrt{B^2 - 4AC}}{2A} \qquad \ldots\ldots (II\text{-}19)$$

which gives always real roots, because the discriminant is positive.
We shall now distinguish the following situations:

1. $B \leq 0$ and $A > 0$ gives as a solution for (II-19)

   $\frac{\alpha}{\lambda} = \frac{-\, B + \sqrt{B^2 - 4AC}}{2A}$, because this is the only root which is positive,

   and the steplength must be positive.

2. $B \leq 0$ and $A < 0$ is impossible. This can be seen as follows. Assume

   $B \leq 0$, then $g_k + z \geq 0$ and $A = g_k + g_k^+ + 2z = g_k + z + g_k^+ + z > 0$

   in branch 1.

   It is impossible to get in branch 2 because when $B \leq 0$, then

   $g_k + z \geq 0$ and in branch 2 $g_k < 0$ and $z < 0$.

3. $B > 0$ and $A > 0$ gives a positive and negative root. The positive root

   is

   $\frac{\alpha}{\lambda} = \frac{-\, B + \sqrt{B^2 - 4AC}}{2A}$ which will be taken because the steplength is

   positive.

4. $B > 0$ and $A < 0$ gives two positive roots.

   The smallest belongs to a minimum the other to a maximum. So we take

the smallest root which is $\frac{\alpha}{\lambda} = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$.

We now take as a solution of the quadratic (II-17)

$$\frac{\alpha}{\lambda} = \frac{-B + \sqrt{B^2 - 4AC}}{2A} = \frac{-B + 2\sqrt{z^2 - g_k g_k^+}}{2A} \equiv \frac{-B + 2P}{2A}.$$

..... (II-20)

Davidon's result can now be checked for the quadratic case.

First we shall prove that

a) $P - g_k - z \neq 0$ and

b) $g_k^+ - g_k + 2P \neq 0$, results that we will need later.

a) $\underline{P - g_k - z \neq 0}$

$A \neq 0$ and $C = g_k < 0 \Rightarrow -B - \sqrt{B^2 - 4AC} \neq 0$ .....(II-21)

$\Rightarrow -B - 2P \neq 0$

$\Rightarrow P - g_k - z \neq 0$ (Q.E.D.)

b) $\underline{g_k^+ - g_k + 2P \neq 0}$ ..... (II-22)

To show this we will distinguish two cases according to the two branches in diagram 1.

1 $g_k < 0$, $g_k^+ > 0$ and now formula (II-22) is obvious.

2 $g_k < 0$, $g_k^+ < 0$

Suppose $g_k^+ - g_k + 2P = 0$, we then have

$$g_k^+ - g_k + 2\sqrt{z^2 - g_k g_k^+} = 0 \Rightarrow$$

$$z^2 - g_k g_k^+ = \frac{g_k^2 + (g_k^+)^2 - 2 g_k g_k^+}{4} \Rightarrow$$

$$z^2 = \frac{g_k^2 + (g_k^+)^2 + 2 g_k g_k^+}{4} \Rightarrow$$

$$z = \pm \frac{g_k + g_k^+}{2}$$ .....(II-23)

But we have shown above that when the process is in branch 2

$$z < \frac{g_k + g_k^+}{2},$$

so (II-22) holds.

Now we proceed verifying Davidon's result. Multiplying the value of (II-20) above and below with the adjoint root you get

$$\frac{\alpha}{\lambda} = \frac{(-B + \sqrt{B^2 - 4AC})\,(-B - \sqrt{B^2 - 4AC})}{2A\,(-B - \sqrt{B^2 - 4AC})} =$$

$$\frac{-2C}{B + \sqrt{B^2 - 4AC}} . \qquad \ldots\ldots (II-24)$$

This can be done because of (II-21). Next, substitution of the values of A, B and C given in (II-17) we obtain

$$\frac{\alpha}{\lambda} = \frac{-g_k}{P - g_k - z} .$$

For the Davidon's results (II-8, II-9) to hold, it remains to verify that

$$1 - \frac{g_k^+ + P - z}{g_k^+ - g_k + 2P} = \frac{-g_k}{P - g_k - z}$$

or

$$\frac{-g_k + z + P}{g_k^+ - g_k + 2P} + \frac{g_k}{P - g_k - z} = 0.$$

We have shown above under ad 2) that a) $P - g_k - z \neq 0$ and b) $g_k^+ - g_k + 2P \neq 0$ so that this can be verified by cross-multiplication taking $P^2 = z^2 - g_k\, g_k^+$.

Computer calculation: Shanno's cubic interpolation procedure.

The cubic interpolation technique devised by Davidon, is used to locate $\alpha_{(r)}$ at each step after two points are found at which $\frac{d\,Q(x;\,\theta)}{d\,\alpha_{(k)}} < 0$ and $\frac{d\,Q(x;\,\theta)}{d\,\alpha_{(k)}} > 0$.

A necessary condition for $\alpha_{(k)}$ to minimize $Q(x; \theta)$ along $Q(\theta^{(k)} + \alpha_{(k)} s^{(k)})$ is that

$$\frac{d\,Q(u)}{d\,\alpha_{(k)}} = \frac{d\,Q(u)}{d\,u} \cdot \frac{du}{d\,\alpha_{(k)}} = g^{(k+1)'} \cdot s^{(k)} = -g^{(k+1)'} H^{(k)} g^{(k)} = 0.$$

As long as $-g^{(k+1)'} H^{(k)} g^{(k)} < 0$ the function $Q(x; \theta)$ is still decreasing and the steplength $\alpha_{(k)}$ must be increased. But after a while $-g^{(k+1)'} H^{(k)} g^{(k)} > 0$ and the function $Q(x; \theta)$ is increasing again. So somewhere in between $-g^{(k+1)'} H^{(k)} g^{(k)} = 0$, and we are searching the $\alpha_{(k)}$ which satisfies this condition.

The computer program uses Shanno's interpolation technique which can be discribed as follows:

Given the functions:

$$S_p(\alpha_j) = a\,\alpha_j^3 + b\,\alpha_j^2 + c\,\alpha_j + d$$

$$S_p(\alpha_{j-1}) = a\,\alpha_{j-1}^3 + b\,\alpha_{j-1}^2 + c\,\alpha_{j-1} + d$$

$$S_p'(\alpha_j) = 3\,a\,\alpha_j^2 + 2\,b\,\alpha_j + c$$

$$S_p'(\alpha_{j-1}) = 3\,a\,\alpha_{j-1}^2 + 2\,b\,\alpha_{j-1} + c\,.$$

The coefficients a, b, c and d can be found as follows

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \alpha_j^3 & \alpha_j^2 & \alpha_j & 1 \\ \alpha_{j-1}^3 & \alpha_{j-1}^2 & \alpha_{j-1} & 1 \\ 3\,\alpha_j^2 & 2\,\alpha_j & 1 & 0 \\ 3\,\alpha_{j-1}^2 & 2\,\alpha_{j-1} & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} S_p(\alpha_j) \\ S_p(\alpha_{j-1}) \\ S_p'(\alpha_j) \\ S_p'(\alpha_{j-1}) \end{pmatrix}$$

The values α_j, α_{j-1}, $S_p(\alpha_j)$, $S_p(\alpha_{j-1})$, $S_p'(\alpha_j)$, $S_p'(\alpha_{j-1})$ are known and by matrix inversion we can find the values for a, b, c and d to solve the quadratic form

$$S_p'(\alpha) = 3\alpha^2 + 2b\alpha + c = 0.$$

That positive valued root is taken for which the function $Q(x; \theta)$ is a minimum.

Appendix III: Numerical results of the Quasi-Newton methods

The methods corresponding to $t = \infty$, $t = \frac{2\,\alpha_{(k)}^{-1}}{\alpha_{(k)}}$, $t = 1$, $t = 0$ and constant norm were tested for various initial estimates on five functions. They are the Weibull function defined by

$$f(\theta_1,\theta_2,\theta_3) = \sum_{i=1}^{99} \left(e^{-\frac{1}{\theta_1}(x_{1i}-\theta_3)^{\theta_2}} - x_{2i}\right)^2 \qquad \ldots\ldots (III\text{-}1)$$

where the x_{2i} and x_{1i} are perfect data generated for the 99 points corresponding to $x_{2i} = .01$ to $.99$ in steps of 0.01 and $x_{1i} = (-50 * \ln x_{2i})^{2/3}+25$. The minimum of this function is zero for the values $\theta_1=50$, $\theta_2=1.5$, $\theta_3=25$. The initial estimates are those suggested by Shanno [19].
The sum of two exponentials documented by Box [2] and defined by

$$f(\theta_1,\theta_2) = \sum_{i=1}^{10} [(e^{-\theta_1 x_i} - e^{-\theta_2 x_i}) - (e^{-x_i} - e^{-10x_i})]^2 \quad \ldots\ldots (III\text{-}2)$$

where x_i ranges from 0.1 to 1 in steps of 0.1, was also tried.
The minimum of the function is zero for the values $\theta_1=1$ and $\theta_2=10$.
The following starting values where used:

(0,0), (0,20), (5,0), (5,20), (2.5,10).

The Rosenbrock function defined by

$f(\theta_1,\theta_2)=100\,(\theta_2-\theta_1^2)^2 + (1-\theta_1)^2$, was also used, with the initial estimates suggested by Leon [14].
The minimum of this function is again zero, for the values $\theta_1=1$ and $\theta_2=1$.
The next function we tried was Wood's function, as documented by Pearson [15] and defined by

$$f(\theta_1,\theta_2,\theta_3,\theta_4) = 100\,(\theta_2-\theta_1^2)^2 + (1-\theta_1)^2 + 90\,(\theta_4-\theta_3^2)^2 +$$

$$+ (1-\theta_3)^2 + 10.1\,[(\theta_2-1)^2 + (\theta_4-\ )^2] + 19.8\,(\theta_2-1)(\theta_4-1). \ldots (III\text{-}4)$$

The initial estimate used was (-3,-1,-3,-1) and the minimum of the function is zero for the vector (1,1,1,1).

The last test function we used was what we shall call the Zangwill function [22] defined by

$$f(\theta_1,\theta_2,\theta_3) = (\theta_1 - \theta_2 + \theta_3)^2 + (-\theta_1+\theta_2+\theta_3)^2 + (\theta_1+\theta_2-\theta_3)^2 \quad \text{....(III-5)}$$

which is a strictly convex quadratic function with a unique minimum at $(\theta_1,\theta_2,\theta_3) = (0,0,0)$.

The initial estimates used were those documented by Zangwill, namely (0.5,1,0.5).

In the table, the number of iterations is the number of times $H^{(k)}$ is updated, and the number of evaluations the actual number of function evaluations used.

In all cases, convergence was said to be reached when for all i

$$\left|\frac{\theta_i^{(k+1)} - \theta_i^{(k)}}{\theta_i^{(k)}}\right| \leq 10^{-5} \quad \text{and} \quad \frac{|g_i^{(k)}|}{|\theta_i^{(k)}|} \leq 10^{-5} .$$

The following versions for selecting t discussed in section 2.4 were tested:

JSCALE = 0 : $t = \infty$

JSCALE = 1 : $t = \dfrac{2\,\alpha_{(k)}^{-1}}{\alpha_{(k)}}$

JSCALE = 2 : t = 1 (Fletcher-Powell)

JSCALE = 3 : t = 0 (Barnes-Rosen)

JSCALE = 4 : constant norm

WEIBULL FUNCTION	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUNCTION EVAL.
	(5,.15,2.5)	x		0	(a)	(a)
		x		1	21	100
		x		2	(a)	(a)
		x		3	20	99
		x		4	19	93
			x	0	19	246
			x	1	25	330
			x	2	31	425
			x	3	20	247
			x	4	24	390
	(250,.3,5)	x		0	(a)	(a)
		x		1	(a)	(a)
		x		2	(b)	(b)
		x		3	38	149
		x		4	34	169

(a) The gradient search has failed

(b) Convergence had not been attained in 100 iterations

(c) " " " " " " 200 "

SUM OF TWO EXPONENTIALS	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUNCTION EVAL.
	(5,20)	x		0	8	37
		x		1	8	34
		x		2	8	38
		x		3	8	33
		x		4	8	34
			x	0	6	119
			x	1	7	136
			x	2	8	128
			x	3	7	107
			x	4	7	156
	(2.5,10)	x		0	6	18
		x		1	6	28
		x		2	6	18
		x		3	6	18
		x		4	6	19
			x	0	5	80
			x	1	5	108
			x	2	5	90
			x	3	5	97
			x	4	5	128

SUM OF TWO EXPONENTIALS	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUCNTION EVAL.
	(0,20)	x		0	10	35
		x		1	10	32
		x		2	10	34
		x		3	10	29
		x		4	10	35
			x	0	7	168
			x	1	7	113
			x	2	8	139
			x	3	7	198
			x	4	7	125
	(5,0)	x		0	15	189
		x		1	15	107
		x		2	15	147
		x		3	18	293
		x		4	(e)	(e)
			x	0	9	111
			x	1	9	135
			x	2	20	255
			x	3	(e)	(e)
			x	4	11	118

(e) overflow has been set

SUM OF TWO EXPONENTIALS	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUNCTION EVAL.
	(0,0)	x		0	11	47
		x		1	12	59
		x		2	11	49
		x		3	11	44
		x		4	11	45
			x	0	10	163
			x	1	10	142
			x	2	10	162
			x	3	13	237
			x	4	10	164

ROSENBROCK FUNCTION	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUNCTION EVAL.
	(1,-1.2)	x		0	55	200
		x		1	30	118
		x		2	55	225
		x		3	62	219
		x		4	26	96
			x	0	18	229
			x	1	25	317
			x	2	18	232
			x	3	15	272
			x	4	17	295
	(-1.2,1)	x		0	14	64
		x		1	16	66
		x		2	14	64
		x		3	14	63
		x		4	15	54
			x	0	21	302
			x	1	24	375
			x	2	22	364
			x	3	23	320
			x	4	19	314

ROSENBROCK FUNCTION	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUNCTION EVAL.
	(2,-2)	x		0	93	310
		x		1	30	115
		x		2	115	487
		x		3	(a)	(a)
		x		4	29	103
			x	0	16	251
			x	1	14	197
			x	2	19	271
			x	3	16	260
			x	4	15	270
	(-3,635,5.621)	x		0	22	80
		x		1	16	67
		x		2	26	122
		x		3	19	68
		x		4	28	113
			x	0	16	183
			x	1	19	264
			x	2	17	222
			x	3	17	238
			x	4	21	207

ROSENBROCK FUNCTION	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUNCTION EVAL.
	(0.639,-0.221)	x		0	29	98
		x		1	27	96
		x		2	33	123
		x		3	29	109
		x		4	23	101
			x	0	22	232
			x	1	20	264
			x	2	17	232
			x	3	21	338
			x	4	17	198
	(1.489,-2.547)	x		0	45	159
		x		1	22	75
		x		2	52	224
		x		3	51	190
		x		4	18	68
			x	0	15	244
			x	1	16	233
			x	2	19	251
			x	3	17	251
			x	4	16	231

WOOD FUNCTION	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF INTERATIONS	NUMBER OF FUNCTION EVAL.
	(-3,-1,-3,-1)	x		0	24	90
		x		1	25	90
		x		2	28	117
		x		3	22	90
		x		4	41	149
			x	0	39	408
			x	1	41	491
			x	2	42	544
			x	3	41	484
			x	4	35	416

ZANGWILL FUNCTION	INITIAL EST.	CUB. INT.	QUADR. INT.	JSCALE	NUMBER OF ITERATIONS	NUMBER OF FUNCTION EVAL.
	(0.5,1,0.5)	x		0	3	22
		x		1	3	20
		x		2	3	21
		x		3	3	20
		x		4	(f)	(f)
			x	0	4	106
			x	1	4	63
			x	2	4	62
			x	3	4	73
			x	4	3	108

(f) no inverse in α-search

Appendix IV: Newton Method

We know that the minimum of a convex function $Q(x;\theta)$ is the solution of the equations

$$\frac{\partial Q(x;\theta)}{\partial \theta_i} = 0, \qquad i = 1,.....p \qquad .....(IV-1)$$

where θ is a p-dimensional parameter vector.

Sometimes the equations are so complicated that iterative methods must be used to find a root, starting from some trial values.
Here we shall show how Newton's method works with one parameter. So we only have one equation to solve:

$$\frac{d\, Q(x;\theta)}{d\ \theta} = 0 \qquad .....(IV-2)$$

This equation can be expanded in a Taylor series, using t as a trial value of θ and retaining only the first power of the derivative with respect to θ

$$0 = \left(\frac{d\, Q(x;\theta)}{d\ \theta}\right)_{\theta = \tilde{\theta}} = \left(\frac{d\, Q(x;\theta)}{d\ \theta}\right)_{\theta = t} + (\tilde{\theta}-t)\left(\frac{d^2 Q(x;\theta)}{d\ \theta^2}\right)_{\theta = \theta^*} \qquad ..... (IV-3)$$

where θ^* is some value in the interval $(\tilde{\theta},t)$, $\tilde{\theta}$ being the exact value of the estimate satisfying (IV-2). Thus

$$\tilde{\theta} = t - \frac{\left(\frac{d\, Q(x;\theta)}{d\ \theta}\right)_{\theta = t}}{\left(\frac{d^2 Q(x;\theta)}{d\ \theta^2}\right)_{\theta = \theta^*}} \qquad .....(IV-4)$$

and we may write approximately that

$$\tilde{\theta} = t - \frac{\left(\frac{d\, Q(x;\theta)}{d\ \theta}\right)_{\theta = t}}{\left(\frac{d^2 Q(x;\theta)}{d\ \theta^2}\right)_{\theta = t}} \qquad .....(IV-5)$$

When, not one but a vector of parameters is to be estimated, it can easily be verified that (IV-5) is identical with

$$\theta^{(k+1)} = \theta^{(k)} - [S^{(k)}]^{-1} g^{(k)}, \qquad \ldots\ldots \text{(IV-6)}$$

$g^{(k)}$ being the vector of first order partial derivatives of $Q(x;\theta)$ with respect to θ_i at $\theta_i^{(k)}$, and $S^{(k)}$ the matrix of second order partial derivatives of $Q(x;\theta)$ with respect to the θ_i's at $\theta_i^{(k)}$.
In formule (IV-6) the unit steplength is used.
In practical problems it often happens that when there is not a good estimate of the position of the minimum of the function which must be minimized, then the iteration (IV-6) is usually worthless.
The drawbacks of iteration IV-6 are

a. it may converge very slowly;

b. it may oscillate widely;

c. it may not converge at all.

Therefore recent papers on minimization algorithms use modifications, so that it is not necessary to be close to the required solution.

Appendix V: Computer program description

Main-program

I. Read statements

a) READ (1,100) INT, INDEX, N, MAX, JSCALE,, EPS, IRIT
100 FØRMAT (I2, I3, 3I5, F15.10, I5)

INT is an integer constant, defining the interpolating method to be used.

We use INT=1 Cubic Interpolation
INT=2 Quadratic Interpolation.

INDEX is an integer constant, indicating what function we are minimizing

= 1 Weibull - function
= 2 Sum of two exponentials (Box)
= 3 Klein-Preston consumption function
= 4 C.E.S. - function additive errors
= 5 C.E.S. - function multiplicative errors
= 6 Zangwill - function
= 7 Rosenbrock - function
= 8 Wood - function

The program is made in such a way to allow other functions to be minimized without much change. Only one subroutine must be elaborated and a new one, which evaluates the function value, must be written (see subroutine EVAL).

N is an integer constant, which is the number of parameters in the function

MAX is an integer constant, which is the maximum number of iterations to be performed. If the calculations have not been terminated for some other reason, they will be stopped when the number of

iterations equals MAX.

JSCALE is an integer constant, indicating the way to evaluate the new Hessian matrix H. See Shanno [19].

JSCALE
- 0 means $t = \infty$
- 1 " $t = \frac{2\alpha - 1}{\alpha}$
- 2 " $t = 1$
- 3 " $t = 0$
- 4 " constant norm version

EPS is a real constant, which is the convergence criterion of the function value and of the parameters

IRIT = 1 means that the values of the variables of the function are on punchcards. In that case there are two more Read Statements:

1) READ (1,100) NT, NN
   100 FØRMAT (I2, I3)
   NT number of observations per variable
   NN number of variables to be read in
2) READ (1,101) XMX
   101 FØRMAT (6F 13.5)
   XMX is a one dimensional vector in which the observations are stored as one long column vector (The stacking operation)
   These statements come after read statement b) see further.

IRIT = 0 in this case the above two read statements are neglected

example:
To make this devise clear take

$\underline{y} = X\beta + \underline{\varepsilon}$ $\quad \underline{\varepsilon} \sim N(0, \sigma^2)$

$X = NT * NN$

$\beta = NN$ vector

$\underline{y} = NT$ vector

if we estimate β and σ^2 using ML, the number of parameters in the likelihoodfunction is N = NN + 1.

b) READ (1,101) (X(I), I = 1,N)

101 FØRMAT (6F 13.5)

X is a real one-dimensional array containing the initial parameter values. New values of X are stored in the vector B.

c) If IRIT = 1 (see above)

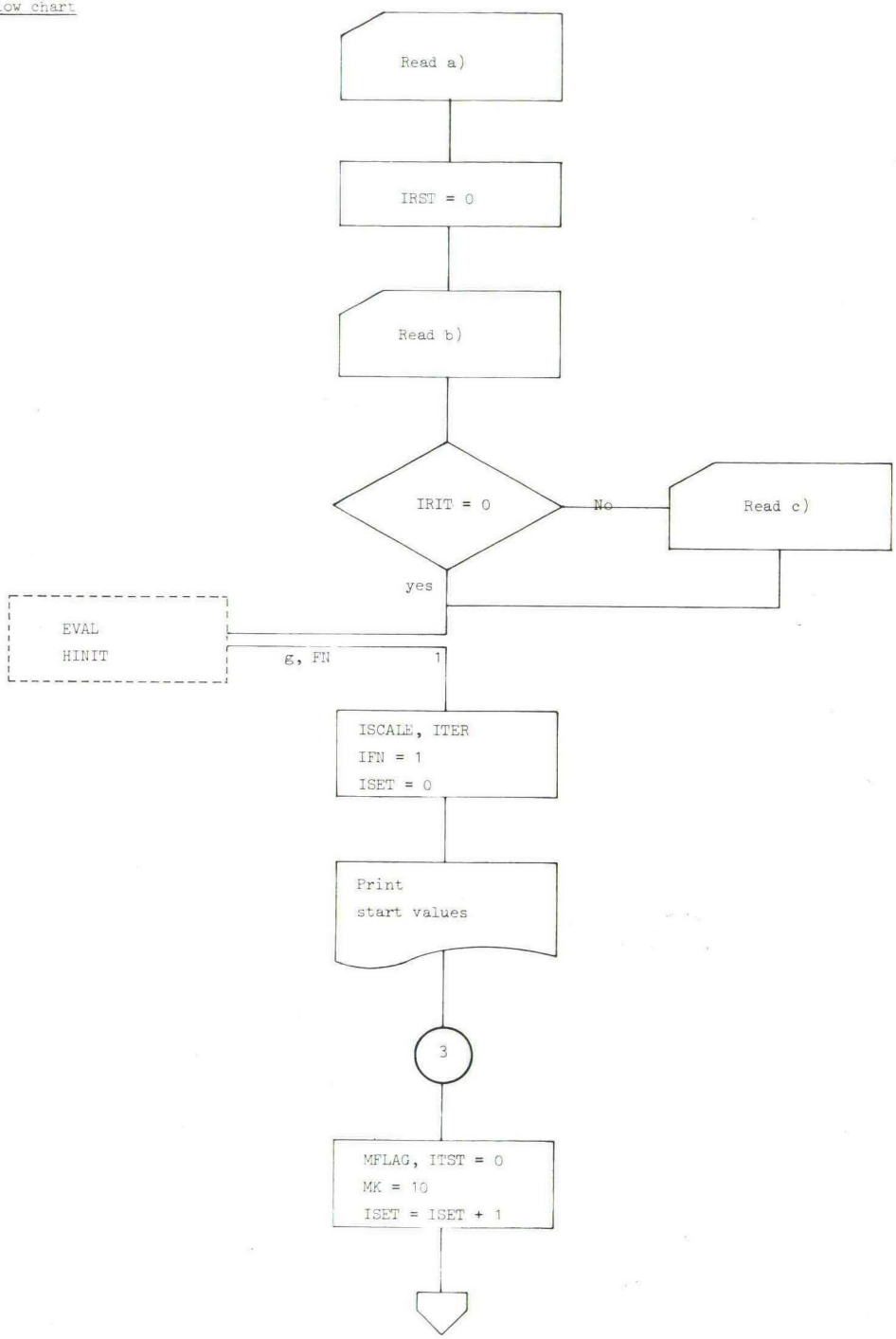
READ (1,100) NT, NN

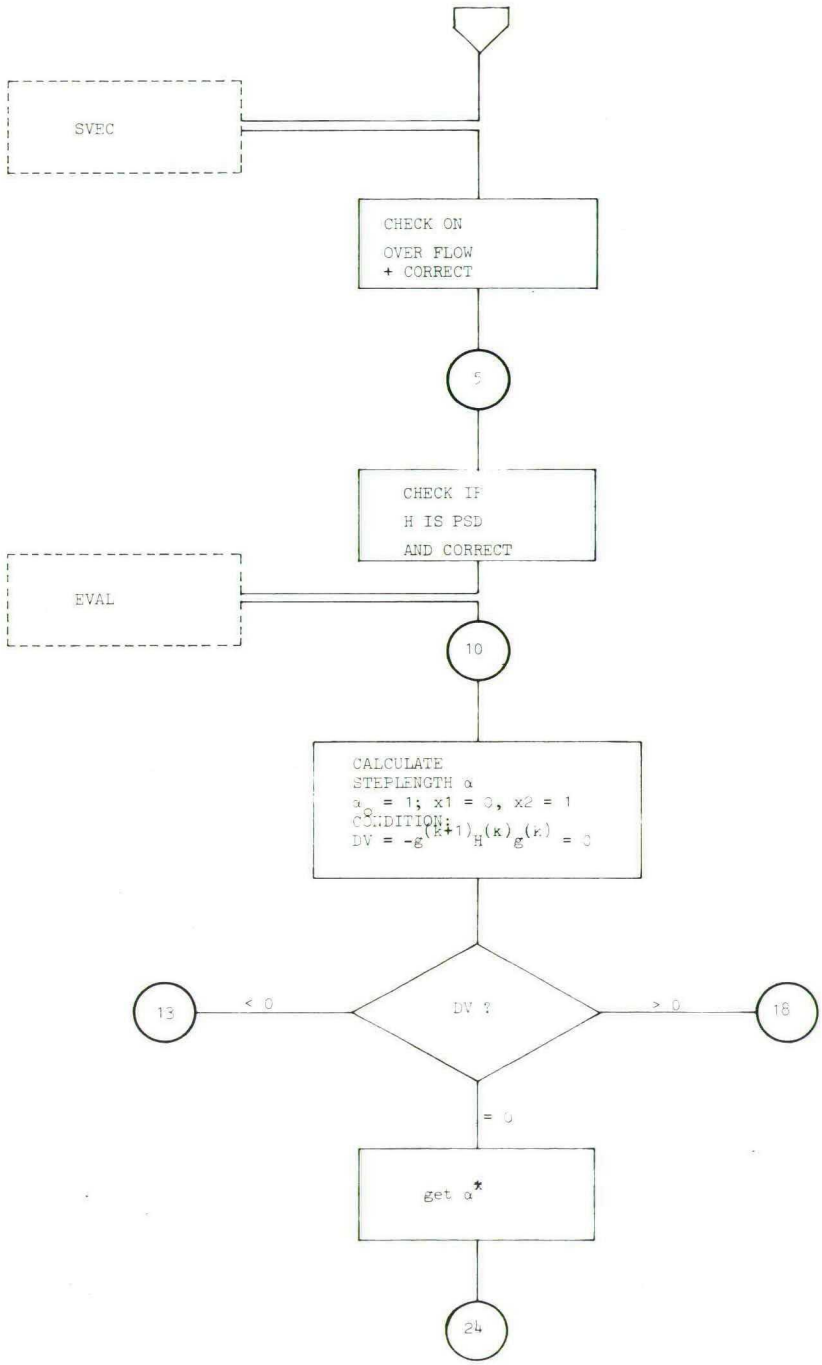
READ (1,101) XMX

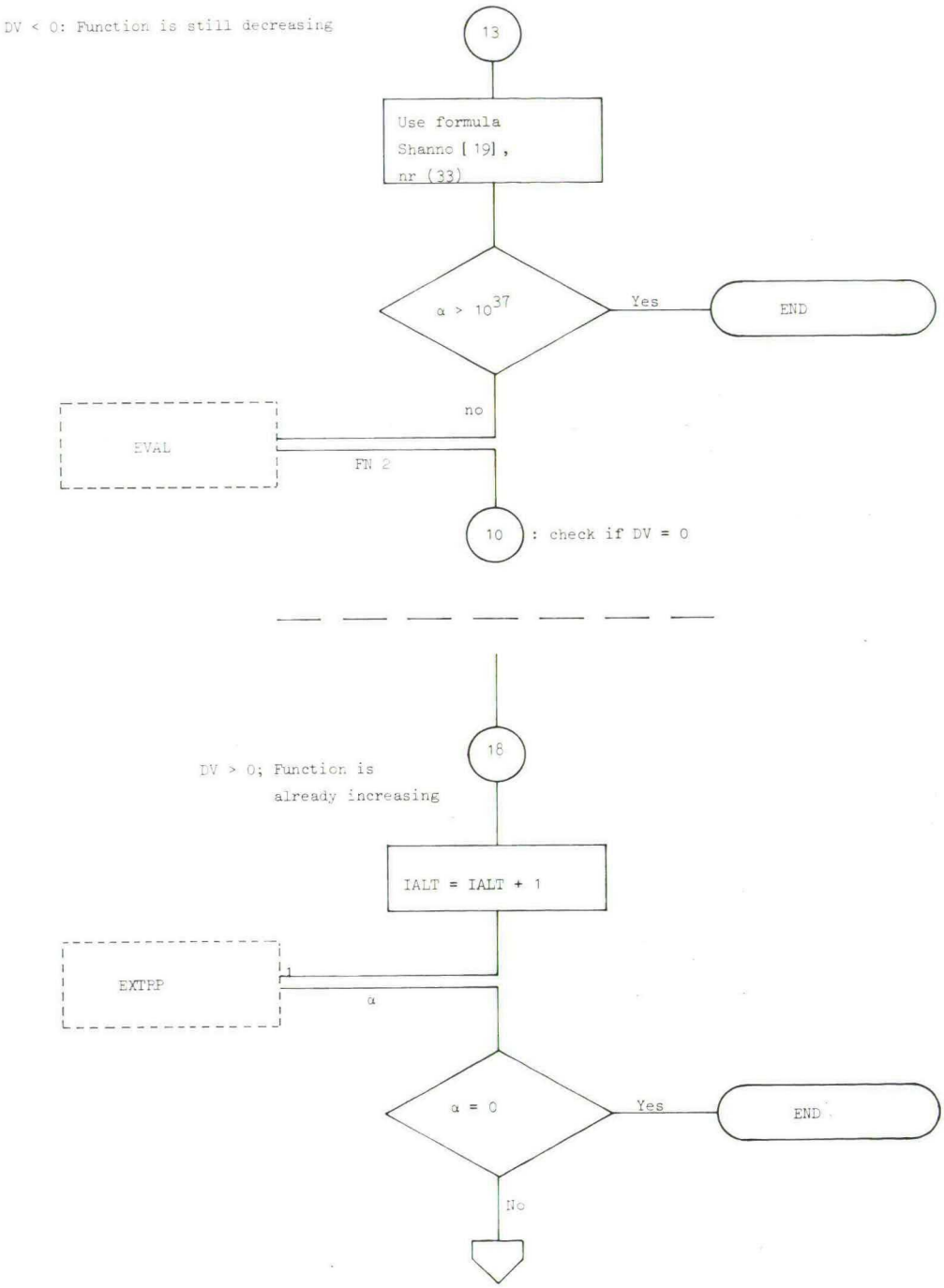
II Explanation of the additional parameters

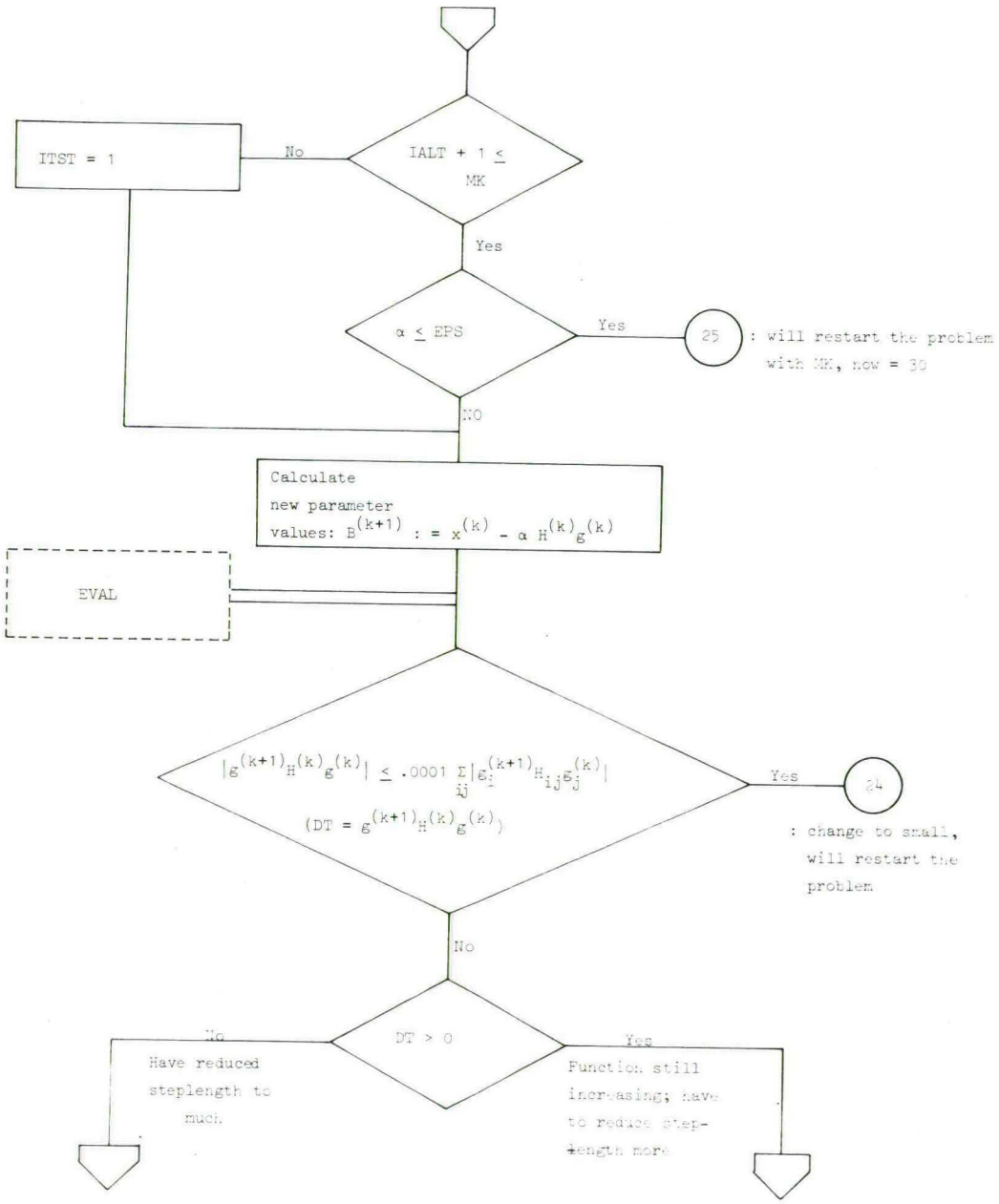
IRST = 0	less than N times a change in the Hessian matrix
= 1	more than N changes
ISET	indicates the number of changes in H (ISET $\leq$ N)
ISCALE = 1	indicates we have rescaled the Hessian matrix after N changes in it: H = I, the identity matrix
= 0	otherwise
ITER	the number of iterations
IFN	number of function evaluations (each time we call EVAL we increase IFN: IFN = IFN + 1)
IALT	number of Davidon-interpolations IALT < MK
IALT2	number of quadratic-interpolations IALT2 < MK
MK	is the maximum number of interpolations, iniatially MK=10 With MFLAG = 1, MK = 30
ITST = 0	as long as IALT or IALT2 < MK
= 1	IALT or IALT2 = MK
MFLAG = 1	this can only happen if at the same time ISCALE = 1. It indicates that we have rescaled the Hessian and restarted the minimization problem completely because the α search is not succesfull. The only change will be a change in the initial parameter startvalues. We now fix MK = 30.
ISCALE = 1	and MFLAG = 0: Because after N iterations we have not yet obtained the minimum of the function, we automatically restart the problem, putting H = I. The difference with if MFLAG = 1 is that the problem is restarted because of unsuccessful minimization in the α-search possibly due to the startvalues of the parameters
FN,FT,FN1,FN2	different function values

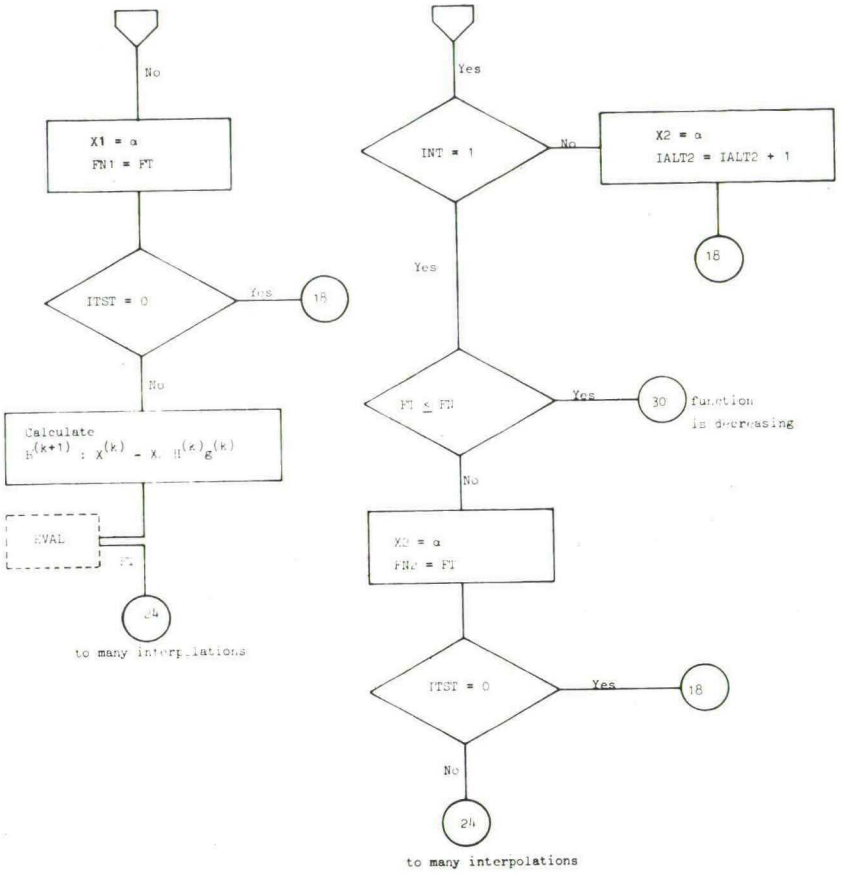
II Flow chart

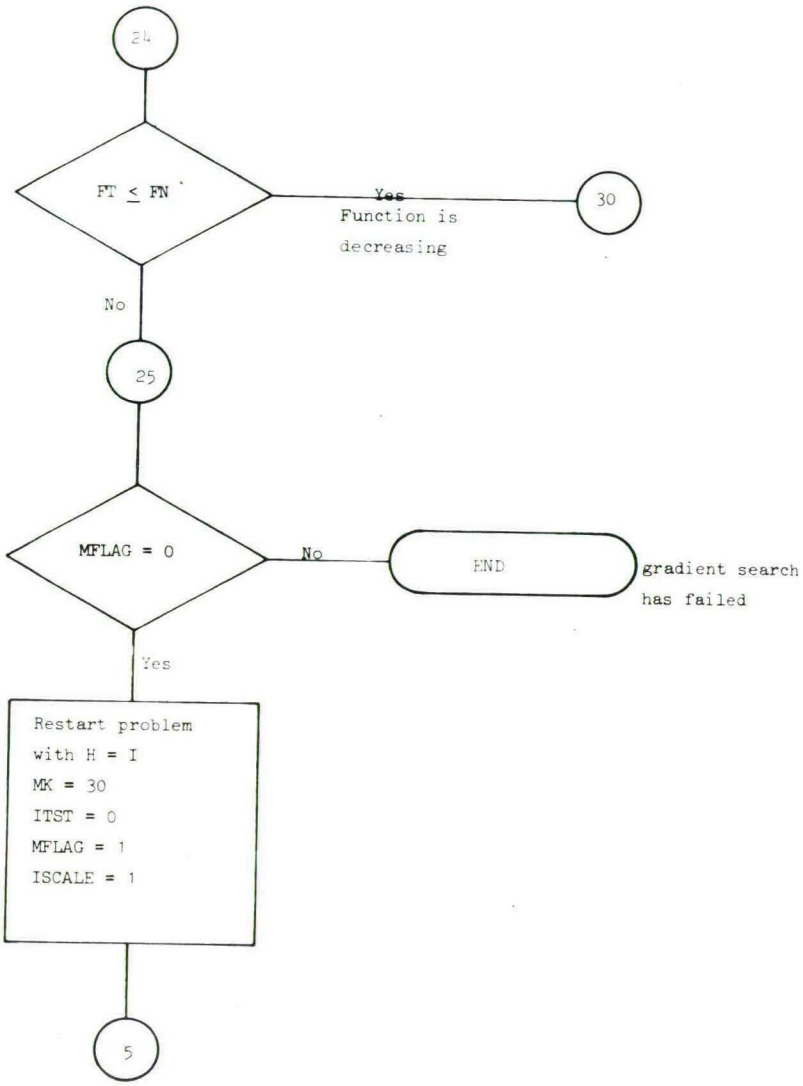


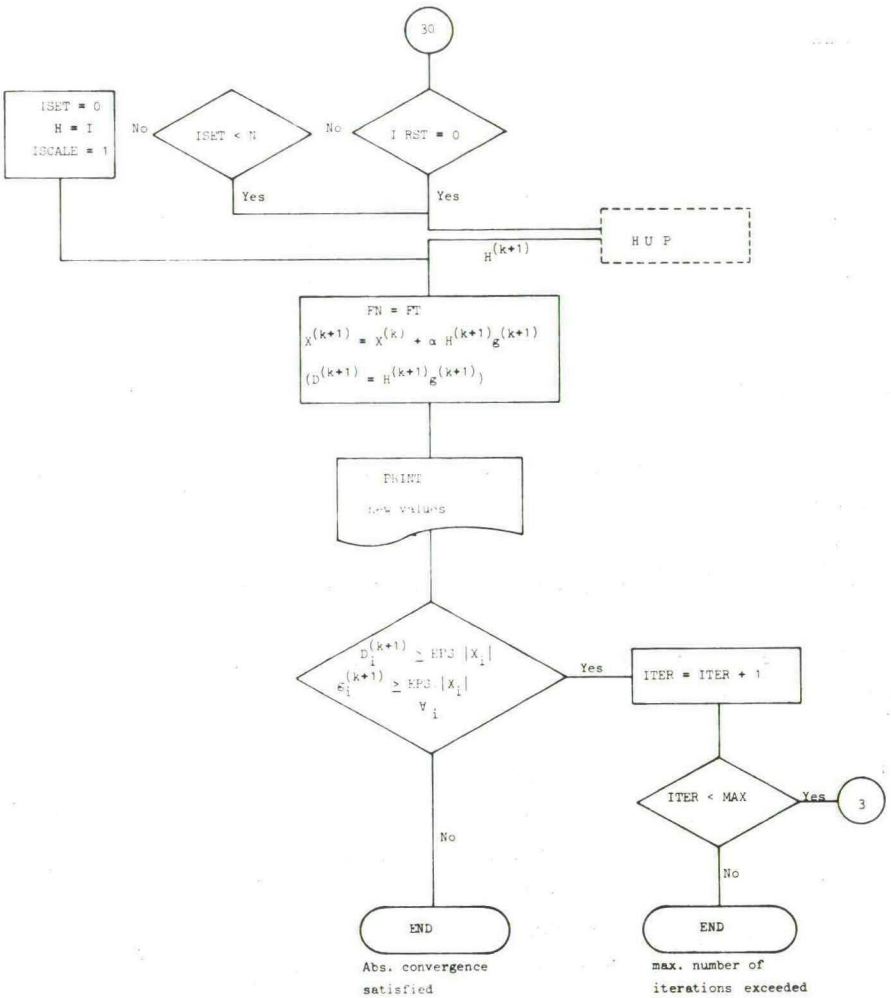

SVEC
CHECK ON
OVER FLOW
+ CORRECT
5
CHECK IF
H IS PSD
AND CORRECT
EVAL
10
CALCULATE
STEPLENGTH α
α0 = 1; x1 = 0, x2 = 1
CONDITION:
DV = -g(k+1) H(k) g(k) = 0
13
< 0
DV ?
> 0
18
= 0
get α*
24


DV < 0: Function is still decreasing
13
Use formula
Shanno [19],
nr (33)
α > 10^37
Yes
END
no
EVAL
FN 2
10
: check if DV = 0
18
DV > 0; Function is
already increasing
IALT = IALT + 1
EXTPP
1
α
α = 0
Yes
END
No


ITST = 1
No
IALT + 1 ≤ MK
Yes
α ≤ EPS
Yes
25
: will restart the problem with MK, now = 30
NO
Calculate new parameter values: B(k+1) : = x(k) - α H(k) g(k)
EVAL
|g(k+1) H(k) g(k)| ≤ .0001 Σij |gi(k+1) Hij gj(k)|
(DT = g(k+1) H(k) g(k))
Yes
24
: change to small, will restart the problem
No
DT > 0
No
Have reduced steplength to much.
Yes
Function still increasing; have to reduce steplength more


No
X1 = α
FN1 = FT
ITST = 0
Yes
18
No
Calculate
$B^{(k+1)} : X^{(k)} - X \cdot H^{(k)} g^{(k)}$
EVAL
FT
24
to many interpolations
Yes
INT = 1
No
X2 = α
IALT2 = IALT2 + 1
18
Yes
FT ≤ FN
Yes
30
function
is decreasing
No
X2 = α
FN2 = FT
ITST = 0
Yes
18
No
24
to many interpolations


24
FT ≤ FN
Yes
Function is
decreasing
30
No
25
MFLAG = 0
No
END
gradient search
has failed
Yes
Restart problem
with H = I
MK = 30
ITST = 0
MFLAG = 1
ISCALE = 1
5


30
ISET = 0
H = I
ISCALE = 1
No
ISET < N
No
I RST = 0
Yes
Yes
H U P
H(k+1)
FN = FT
X(k+1) = X(k) + α H(k+1) g(k+1)
(D(k+1) = H(k+1) g(k+1))
PRINT
new values
Yes
ITER = ITER + 1
ITER < MAX
Yes
3
No
No
END
Abs. convergence satisfied
END
max. number of iterations exceeded

III .Subroutines

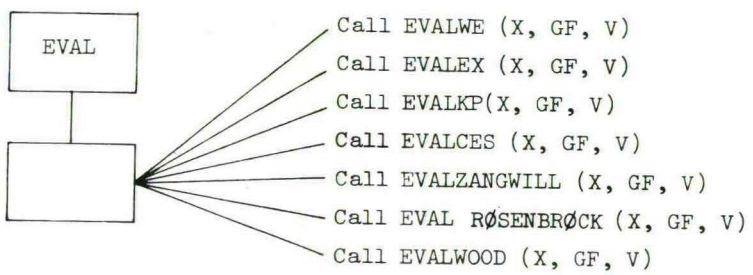
1. Subroutine EVAL

Since the value of the object function is needed at various stages during the program as well as the value of the gradient vector, a subroutine is provided to compute these values. Whenever these values are needed for a particular value of the parameters the main program transmits the vector of current values of X to that subroutine called EVAL.

CALL EVAL (X, GF, V)

X is a one dimensional array containing the value of the current estimates to the solution
GF is the gradient vector ∇f at X
V is a real variable indicating the new value of the function

The function of EVAL is not in itself to calculate the above information, but to call another subroutine.



These subroutines refer to different functions (see Appendix III)

Also available in CØMMØN are

1. INDEX — constant indicating which subroutine to select in the function evaluation
2. N — number of parameters in the function
3. XMX, NT, NN — as above.
   For certain functions (e.g. Weibull-function), the observations are generated in the subroutine itself, whereas for others (e.g.) Klein-Preston consumption function, CES-function), time series are used (see also Mainprogram : IRIT=1).

2. Subroutine HINIT

This subroutine calculates the value of the initial Hessian matrix.

CALL HINIT (H, N)

N as above

H is a N X N identity matrix, used as a starting value of the Hessian matrix

3. Subroutine SVEC

The SVEC subroutine calculates the direction, in which the function can be minimized.

$$D^{(k)} = -H^{(k)}g^{(k)}$$

CALL SVEC (H, GF, D, N)

H, N, GF as above

D is the real N dimensional vector containing the coordinates of the direction

4. Subroutine HUP

This subroutine is based on Shanno [19] and Shanno and Kettler [20]. The purpose is to update the Hessian matrix, where the particular way of doing this is based on the value of JSCALE.

CALL HUP (N, JSCALE, ALPHA, D, G, C, H, T)

N, JSCALE as above

ALPHA is the optimal steplength

D is the real N dimensional vector with the coordinates of the direction at X^k ($D=-H^{(k)}g^{(k)}$)

G the gradient of the function at the previous parameter values: i.e. at $X^{(k)}$

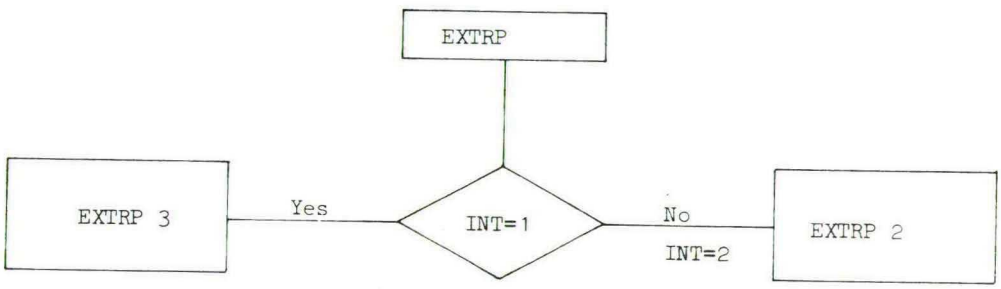
C the gradient of the function at $X^{(k+1)}$

H the Hessian matrix. It transfers the Hessian at the values of $X^{(k)}$ to the subroutine and after updating it, stores the new matrix under the same name.

T — a real constant transfering to the main program the value of t, used in updating the Hessian matrix (See Shanno [19]). This is done only for information purposes.

5. Subroutine EXTRP

This subroutine calculates the optimal steplength using Davidon's method (cubic interpolation) or a quadratic interpolation technique. It is only an intermediate program which CALL EXTRP3 or EXTRP2 depending on the parameter INT.



CALL EXTRP (X1, X2, F1, F2, DF1, DF2, ALPHA, INT)

All parameters will be explained below in EXTRP3.

5a. Subroutine EXTRP3

This subroutine calculates the optimal steplength using the Davidon cubic-interpolation procedure [6]. The formulas used are based on a program made available to us by Mr.D.F.Shanno. An explanation of this is also given in Appendix II.

CALL EXTRP3 (X1, X2, F1, F2, DF1, DF2, ALPHA)

X1, X2, — real variables between which the function has to be interpolated for finding minimum value of it

F1, F2	functionvalue respectively at X1 and at X2
DF1, DF2	value of the derivative at X1, at X2
ALPHA	optimal variable value for which the function is minimum

5b. Subroutine EXTRP2

The subroutine uses a quadratic interpolation technique for minimizing a function of a single variable. See Appendix I.

CALL EXTRP2 (X1, X2, F1, F2, DF1, DF2, ALPHA)

All parameters are as in EXTRP3.

```
      MASTER E211
C
C QUASI-NEWTON METHOD/SHANNO EVALUATION
C
C INT = 1 : CUBIC INTERPOLATION
C       2 : QUADRATIC INTRPOLATION
C INDEX = 1 : WEIBULL FUNCTION
C       = 2 : SUM OF TWO EXPONENTIALS (BOX)
C       = 3 KLEIN-PRESTON CONSUMPTION FUNCTION
C       = 4 C.E.S.-FUNCTION ADDITIVE ERRORS
C       = 5 C.E.S.-FUNCTION MULTIPLICATIVE ERRORS
C       = 6 ZANGWILL FUNCTION
C       = 7 ROSENBROCK FUNCTION
C       = 8 WOOD FUNCTION
C
C JSCALE = 0 : T = ONEINDIG
C        = 1 : T = (2*ALPHA-1)/ALPHA
C        = 2 : T = 1
C        = 3 : T = 0
C        = 4 : CONSTANT NORMVERSION
C
      LOGICAL WWW
C
  100 FORMAT ( I2,I3,3I5,F15.10,2I5,F15.4,I5)
  101 FORMAT (6F13.5)
  102 FORMAT (15H THE INITIAL F=E16.8,5H MAX=15,5H EPS=F10.8,8H JSCALE=I
```

```
    13,7H INDEX=13)
103 FORMAT ( 33H THE VALUE OF THE PARAMETERS IS //)
104 FORMAT (1H 7E16.8)
105 FORMAT (22H THE GRADIENT OF F IS //)
106 FORMAT (21H THE HESSIAN OF F IS //)
107 FORMAT (1H ,20(4H----))
108 FORMAT (9H ITER.NO I5,2H  I5,13H FN.EVALS. F=E16.8//7H ALPHA=E16.8
   1,7H MFLAG=12,3H T=E14.6)
109 FORMAT (24H THE CURRENT DELTA X IS //)
110 FORMAT (48H MAX.ITERATIONS EXCEEDED. EXECUTION TERMINATED.)
111 FORMAT (29H THE ALGORITHM HAS CONVERGED.)
112 FORMAT (I5)
113 FORMAT (50H ALPHA GREATER THAN 10.**37.EXECUTION TERMINATED.)
114 FORMAT (52H A GRADIENT SEARCH HAS FAILED EXECUTION TERMINATED.)
115 FORMAT (47H DAVIDON-FLETCHER-POWELL OPTIMIZATION TECHNIQUE)
116 FORMAT (19H INFINITE T VERSION)
117 FORMAT (19H SCALED F-P VERSION)
118 FORMAT (23H FLETCHER POWELL METHOD)
119 FORMAT (20H BARNES ROSEN METHOD)
120 FORMAT (22H CONSTANT NORM VERSION)
121 FORMAT (18H ALPHA IS NEGATIVE)
122 FORMAT (1H ,9F13,8)
123 FORMAT (1H1)
125 FORMAT (1H ,//)
127 FORMAT (25H CONSTRAINED OPTIMIZATION)
```

```
C
      DIMENSION B(20),C(20),D(20),G(20),H(20,20), XMX(500), X(20), BD(20),
     1ITP(20),BB(20),CC(20)
C
      COMMON /EV/INDEX,N,IPR/EQ/XMX,NT,NN,B,C,/WE/IWARN
      COMMON /E2/D,X,BD,IFN
      COMMON /YQ/BB,CC
C
    1 READ (1,100) INT,INDEX,N,MAX,JSCALE,EPS,IRIT,ITRANS,TRANS,ITRANSP


      IRST,IWARN=0
      IPR=1
      WRITE (2,125)
      WRITE (2,115)
      WRITE (2,125)
  204 DELTA=0.001
      READ (1,101) (X(I),I=1,N)
      IF (ITRANSP.EQ.0) GO TO 321
      READ (1,101) (BB(I),I=1,N)
      READ (1,101) (CC(I),1=1,N)
      WRITE (2,127)
      WRITE (2,125)
  321 IF (IRIT.EQ.0) GO TO 200
      READ (1,100) NT,NN
      J2=0
      DO 201 I=1,NN
```

```
      J1=J2+1
      J2=J1+NT-1
      READ (1,101) (XMX(J),J=J1,J2)
      IF(ITRANS)300, 305,300
  300 DO 202 J=J1,J2
      XMX (J) = XMX(J)/TRANS
  202 CONTINUE
  305 WRITE (2,112) I
  201 WRITE (2,122) (XMX(J),J=J1,J2)
  200 CALL EVAL (X,G,FN)
      CALL HINIT (H,N)
      ISCALE,ITER,IFN=1
      ISET=0
      IF(JSCALE.GE.1) GO TO 56
      WRITE (2,116)
      GO TO 54
   56 GO TO (0,70,72,73),JSCALE
      WRITE (2,117)
      GO TO 54
   70 WRITE (2,118)
      GO TO 54
   72 WRITE (2,119)
      GO TO 54
   73 WRITE (2,120)
   54 WRITE (2,125)
```

```
      WRITE (2,102) FN, MAX, EPS, JSCALE, INDEX
      WRITE (2,103)
      WRITE (2,104) (X(I), I=1,N)
      WRITE (2,105)
      WRITE (2,104) (G(I), I=1,N)
      WRITE (2,106)
      DO 2 I=1,N
    2 WRITE (2,104) (H(I,J), J=1,N)
      WRITE (2,107)
    3 MFLAG, ITST=0
      MK=10
      ISET=ISET+1
    4 XLAM=0.0
      CALL SVEC (H,G,D,N)
      SUMP=0.0
C
      DO 88 I=1,N
   88 SUMP=SUMP-D(I)*G(I)
C
    5 DV=0.
      LTST=0
      DO 6 I=1,N
      DV=DV+G(I)*D(I)
```

```
 6 B(I)=X(I)+D(I)
   IF (DV.LE.0) GO TO 14
97 DO 98 I=1,N
98 D(I)=-D(I)
   SUMP=-SUMP
   WRITE (2,121)
   GO TO 5
14 CALL EVAL (B,C.FN2)
   IFN=IFN+1
   IF (IWARN.LE.0) GO TO 9
 7 DO 8 I=1,N
 8 D(I)=D(I)/2.0
   DO 62 I=1,N
   DO 62 J=1,N
62 H(I,J)=H(I,J)+2.0*D(I)*D(J)/DV
   GO TO 5
 9 X1=0.0
   X2=1.0
   FN1=FN
   DV1=DV
10 DV2=0.0
   DO 11 I= 1,N
11 DV2=DV2+D(I)*C(I)
   IF (DV2) 13,12,47
```

```
12  FT=FN2
    ALPHA=X2
    GO TO 24
13  IF (LTST.EQ.0) GO TO 63
    X1=X2
    X2=(XPERM+X2)/2.0
    GO TO 65
63  X1=X2
    XTRY=SUMP/(2.0*FN2-FN1+SUMP))
    IF (XTRY.LE.X2) GO TO 81
    X2=XTRY
    GO TO 65
81  X2=2.0*X2
65  FN1=FN2
    DV1=DV2
    IF (X2.GT.1.0E37) GO TO 17
15  DO 16 I=1,N
16  B (I)=X(I)+D(I)*X2
    CALL EVAL (B,C,FN2)
    IFN=IFN+1
    IF (IWARN.LE.0) GO TO 10
    XPERM=X2
    X2=(X1+X2)/2.0
    LTST=1
    GO TO 15
```

```
   17 WRITE (2,113)
      GO TO 38
C
   47 IALT = 1
   18 CALL EXTRP (X1,X2,FN1,FN2,DV1,DV2, ALPHA,INT)
      IF (ALPHA.LT.0.) GO TO 216
      IF (ALPHA.EQ.0) GO TO 38
      IALT=IALT+1
      IF (IALT.LE.MK) GO TO 48
      ITST=1
      GO TO 40
   48 IF (ALPHA.LE.EPS) GO TO 25
   40 DO 19 I=1,N
   19 B(I)=X(I)+ALPHA*D(I)
      CALL EVAL (B,C,FT)
      IFN=IFN+1
      DT,TST=0.0
      DO 20 I=1,N
      TST=TST+ABS(C(I)*D(I))
   20 DT=DT+C(I)*D(I)
      IF (ABS (DT).LE.DELTA*TST) GO TO 24
      IF (DT.GT.0) GO TO 23
      X1=ALPHA
      FN1=FT
      DV1=DT
```

```
      IF(ITST.EQ.0) GO TO 18
      DO 51 I=1,N
   51 B(I)=X(I)+X2*D(I)
      ALPHA = X2
      CALL EVAL (B,C,FT)
      IFN=IFN+1
      GO TO 24
   23 IF(FT.LE.FN) GO TO 30
      X2=ALPHA
      FN2=FT
      DV2=DT
      IF(ITST.EQ.0) GO TO 18
   24 IF(FT.LE.FN) GO TO 30
   25 IF(MFLAG.GT.0) GO TO 29
      S=0.0
      DO 93 I=1,N
   93 S=S+G (I)*G(I)
      S=SQRT (S)
      DO 28 I=1,N
      D(I)=-G(I)/S
      DO 27 J=1,N
   27 H(I,J)=0.0
   28 H(I,I)=1.0
      MK=30
      ITST=0
```

```
     MFLAG,ISCALE=1
     GO TO 5
29   WRITE (2,114)
     GO TO 38
30   IF (IRST.EQ.0) GO TO 44
41   IF (ISET.LT.N) GO TO 44
46   ISET=0
     DO 43 I=1,N
     DO 42 J=1,N
42   H(I,J)=0.0
43   H(I,I)=1.0
     ISCALE =1
     GO TO 45
44   CALL HUP (N,JSCALE, ALPHA,D,G,C,H,TC)
45   FN=FT
     DO 31 I=1,N
     D(I)=ALPHA*D(I)
31   X(I)=X(I)+D(I)
     WRITE (2,108) ITER,IFN,FN,ALPHA,MFLAG,TC
     WRITE (2,103)
     WRITE (2,104) (X(I),I=1,N)
     WRITE (2,105)
     WRITE (2,104) (C(I),I=1,N)
     WRITE (2,109)
     WRITE (2,104) (D(I),I=1,N)
```

```
      WRITE (2,106)
      DO 32 I=1,N
   32 WRITE (2,104) (H(I,J),J=1,N)
      WRITE (2,107)
C
      DO 34 I=1,N
      IF (ABS(D(I)).GT.EPS*ABS(X(I)+.001))go to 35
      IF (ABS(G(I)).GT.EPS*ABS(X(I)+001) )go to 35
   34 CONTINUE
      WRITE (2,111)
      GO TO 38
   35 DO 36 I=1,N
   36 G(I)=C(I)
      ITER=ITER+1
  250 IF (ITER.LE.MAX) GO TO 3
      WRITE (2,110)
C
   38 READ (1,112) ICONT
      WRITE (2,123)
      GO TO 217
  216 WRITE (2,121)
  217 IF (ICONT.NE.0) GO TO 1
      STOP 99999
      END

END OF SEGMENT, LENGTH 1487, NAME E211
```

```
C
      SUBROUTINE HINIT (H,N)
C
      DIMENSION H(20,N)
C
      DO 2 I=1,N
      DO 1 J=1,N
    1 H(I,J)=0.0
    2 H(I,I)=1.0
      RETURN
      END
```

END OF SEGMENT, LENGTH 53, NAME HINIT


```
C
      SUBROUTINE SVEC (H,G,D,N)
C
      DIMENSION H(20,N),G(N),D(N)
C
      DO 1 I=1,N
      D(I)=0.
      DO 1 J=1,N
    1 D(I)=D(I)-H(I,J)*G(J)
      RETURN
      END
```

END OF SEGMENT, LENGTH 87, NAME SVEC

```
C
      SUBROUTINE EXTRP (X1,X2,F1,F2,DF1,DF2,ANS,INT)
C
      GO TO (1,2),INT
    1 CALL EXTRP3 (X1,X2,F1,F2,DF1,DF2,ANS)
      GO TO 100
    2 CALL EXTRP2 (X1,X2,F1,F2,DF1,DF2,ANS)
  100 RETURN
      END
```

END OF SEGMENT, LENGTH 35, NAME EXTRP

```
C
      SUBROUTINE EXTRP2 (X1,X2,F1,F2,DF1,DF2,ANS)
C
      DIMENSION X(20),D(20),BD(20),C3(5)
C
      COMMON /EV/INDEX,N,IPR
      COMMON /E2/D,X,BD,IFN
C
  212 FORMAT (26H ER ZIJN MINSTENS 2 MINIMA)
  128 FORMAT (15H NIETS GEVONDEN)
      ANS=0.
      D1=X2-X1
      DO35 I=1,50
      D1=0,5*D1
      X3=X1+D1
      DO 600 J=1,N
```

```
  600 BD(J)=X(J)+X3*D(J)
      CALL EVAL (BD,C3,F3)
      IFN=IFN+1
      IF(F3.GT.F1) GO TO 35
      IF(F2+F1.GT.2*F3) GO TO 900
      IF(I.EQ.1.) WRITE (2,212)
   35 X2=X3
      WRITE (2,128)
      ANS=-10.
      RETURN
  900 ANS=(X1+X3)/2.-D1*(F3-F1)/(F2-2.*F3+F1)
      RETURN
      END
  END OF SEGMENT, LENGTH  331, NAME  EXTRP2

C
      SUBROUTINE EXTRP3 (X1,X2,F1,F2,DF1,DF2,ANS)
C
   20 FORMAT (40H NO INVERS IN ALPHA-SEARCH (SUBR. EXTRP))
C
      DIMENSION A(4,4),BB(4),BC(4),IZ(4)
C
      IF (X1.NE.0) GO TO 3
      A(1,1),A(1,2),A(1,3),A(3,1),A(3,2)=0.0
      DO 11 I=1,3
```

```
 11 A(2,I)=X2**(4-I)
    DO 22 I=1,2
    EI=4-I
 22 A(4,I)=EI*X2**(3-I)
    GO TO 15
  3 DO 1 I=1,3
    A(1,I)=X1**(4-I)
  4 A(2,I)=X2**(4-I)
    DO 2 I=1,2
    EI=4-I
    A(3,I)=EI*X1**(3-I)
  2 A(4,I)=EI*X2**(3-I)
 15 A(1,4),A(2,4),A(3,3),A(4,3)=1.0
    A(3,4),A(4,4)=0.0
    DET=10E-20
    N=4
    CALL INVERT (A,4,N,BB,BC,IZ,DET)
    IF (N.GT.O) GO TO 510
    WRITE (2,20)
    ANS=0.0
    RETURN
510 BC(1)=F1
    BC(2)=F2
    BC(3)=DF1
    BC(4)=DF2
```

```
      DO 10 I=1,4
      BB(I)=0.0
      DO 10 J=1,4
   10 BB(I)=BB(I)+A(I,J)*BC(J)
      DISC=SQRT(ABS(4.*BB(2)*BB(2)-12.0*BB(1)*BB(3)))
      IF (ABS(BB(1)).GT..0001*DISC) GO TO 9
      ANS=-BB(3)/(2.0*BB(2))
      GO TO 7
C
C
    9 T1=(-2.0*BB(2)-DISC)/(6.0*BB(1))
      T2=(-2.0*BB(2)+DISC)/(6.0*BB(1))
      IF (BB(1).GT.o) GO TO 5
      IF (T2.LT.X1) GO TO 6
    4 ANS=T2
      GO TO 7
    5 IF (T1.LT.X1) GO TO 4
    6 ANS=T1
    7 RETURN
      END
```

END OF SEGMENT, LENGTH 473, NAME EXTRP3

```
C
      SUBROUTINE HUP (N,JSCALE,ALPHA,D,G,C,H,T)
C
      DIMENSION  D(N),G(N),C(N),H(20,N),B(20),E(20)
C
      IF(JSCALE-1) 16,1,0
      GO TO (1,0,9,1),JSCALE
      T=1.0
      GO TO 12
    1 TSCALE,T2=0.0
      DO 3 I=1,N
      TSCALE=TSCALE-D(I)*G(I)
      E(I)=0.0
      DO 2 J=1,N
    2 E(I)=E(I)+H(I,J)*C(J)
    3 T2=T2+C(I)*E(I)
      T1=ALPHA*TSCALE
      DO 4 I=1,N
      DO 4 J=1,N
    4 H(I,J)=H(I,J)+(ALPHA-1.0)*D(I)*D(J)/TSCALE
      IF (JSCALE-2) 0,6,7
      T=1.0
      GO TO 12
    6 T=ALPHA-1.0+1.0/ALPHA
      GO TO 12
```

```
 7 S1,S2=0.0
   DO 8 I=1,N
   S1=S2+ALPHA*ALPHA*D(I)*D(I)
 8 S2=S2+(T1*E(I)-T2*D(I)**2
   IF (JSCALE.GT.3) GO TO 10
 9 S=0
   GO TO 11
10 S=SQRT(S1)/SQRT(S2)
11 T=T2*S/(1.0-T1*S)
   IF (T.LT.0)GO TO 21
12 S1,S2=0.0
   DO 14 I=1,N
   S1=S1+ALPHA*D(I)*(C(I)-G(I))
   B(I)=0.
   DO 13 J=1,N
13 B(I)=B(I)+H(I,J)*(C(J)-G(J))
14 S2=S2+((1.0-T)*ALPHA*D(I)-B(I))*(C(I)-G(I))
   DO 15 I=1,N
   DO 15 J=I,N
15 H(I,J)=H(I,J)+T*ALPHA*ALPHA*D(I)*D(J)/S1+((1.0-T)*ALPHA*D(I)-B(I))
  1*((1.0-T)*ALPHA*D(J)-B(J))/S2
   RETURN
C
16 T=1.0E35
   T1,T2=0.0
```

```
      DO 18 I=1,N
      B(I)=0.0
      DO 17 J=1,N
   17 B(I)=B(I)+H(I,J)*(C(J)-G(J))
      T1=T1+(C(I)-G(I)*B(I)
   18 T2=T2+ALPHA*D(I)*(C(I)-G(I))
      R=T2/(T1+T2)
      T2=0.
      DO 19 I=1,N
   19 T2=T2+(ALPHA*D(I)-R*B(I))*(C(I)-G(I))
      DO 20 I=1,N
      DO 20 J=1,N
   20 H(I,J)=H(I,J)+(R-1.0)*B(I)*B(J)/T1+(ALPHA*D(I)-R*B(I))*(ALPHA*D(J)
     1-R*B(J))/T2
      RETURN
C
   21 DO 22 I=1,N
      DO 22 J=1,N
   22 H(I,J)=H(I,J)+(1.0-ALPHA)*D(I)*D(J)/TSCALE
      GO TO 16
      END

END OF SEGMENT, LENGTH   865, NAME  HUP
```

```
C
C
      SUBROUTINE EVALWE (X,DX,V)
C
      DIMENSION X(3),DX(3),XM(3)
C
      COMMON /WE/IWARN
C
      IWARN=0
      Y,V=0.0
      DO 1 I=1,3
    1 DX(I)=0.0
      DO 2 I=1,99
      Y=Y+0.01
      Z=(-50.0*ALOG(Y)**(.66666666667)+25.0
      IF (X(2).GE.10.0) GO TO 4
    5 T2=(-1.0/X(1))*(ABS(Z-X(3)))**X(2)
      IF (T2.LE.20.0) GO TO 3
    4 IWARN=1
      RETURN
    3 S1=EXP(T2)
      S=S1-Y
      S1=S1*T2
      XM(1)=-S1/X(1)
      XM(2)=S1*ALOG(ABS(Z-X(3)))
      XM(3)=-X(2)/(Z-X(3))*S1
      DO 6 J=1,3
```

```
    6 DX(J)=DX(J)+XM(J)*S*2.0
      XM(1),XM(2),XM(3)=0.0
    2 V=V+S*S
      RETURN
      END
END OF SEGMENT, LENGTH   254, NAME  EVALWE


C
      SUBROUTINE EVALEX (X,DX,V)
C
      DIMENSION X(2),DX(2),XM(2)
C
      Y,V=0.0
      DO 1 I=1,2
    1 DX(I)=0.0
      DO 2 I=1,10
      Y=Y+0.1
      Z=EXP(-X(1)*Y)-EXP(-X(2)*Y)-EXP(-Y)+EXP(-10.0*Y)
      XM(1)=-Y*EXP(-X(1)*Y)
      XM(2)=Y*EXP(-X(2)*Y)
      DO 6 J=1,2
    6 DX(J)=DX(J)+XM(J)*Z*2.0
      XM(1),XM(2)=0.0
    2 V=V+Z*Z
      RETURN
```

```
      END
END OF SEGMENT, LENGTH   193, NAME  EVALEX


C
      SUBROUTINE EVALKP (X,DX,V)
C
      DIMENSION X(5),DX(5),XMX(500)
C
      COMMON /EQ/XMX,NT,NN
C
      PHI=3.1415926535
      V=-(ALOG(2.*PHI)+ALOG(X(1)))*NT/2.
      DX(1)=-NT/(2.*X(1))
      DO 100 I=2,5
  100 DX(I)=0.0
      DO 101 I=1,NT
      J=NT+I
      ETH=EXP(-X(5)*XMX(J))
      Z1=1.+X(4)*ETH
      Z2=Z1*Z1
      Z=XMX(I)-X(2)-X(3)/Z1
      ZZ=Z*Z
      V=V-ZZ/(2.*X(1))
      DX(1)=DX(1)+ZZ/(2.*X(1)*X(1))
      DX(2)=DX(2)+Z/X(1)
      DX(3)=DX(3)+Z/(Z1*X(1))
```

```
      DX(4)=DX(4)-(Z*X(3)*ETH)/(Z2*X(1))
  101 DX(5)=DX(5)+(Z*X(3)*X(4)*ETH)/(Z2*X(1))
      V=-V
      DO 102 I=1,5
  102 DX(I)=-DX(I)
      RETURN
      END
END OF SEGMENT, LENGTH   351, NAME  EVALKP


C
      SUBROUTINE EVALCES (X,DX,V)
C
      INTEGER T
C
C
      DIMENSION X(5),DX(5),XMX(500)
C
      COMMON /EQ/XMX,NT,NN
C
      V=0.0
      DO 100 I=1,5
  100 DX(I)=0.0
      Z3=-X(5)/X(4)
      DO 101 I=1,NT
      K=NT+I
      L=2*NT+I
      T=3*NT+I
      Z2=EXP(X(2)*XMX(T))
```

```
      Z5=X(1)*Z2
      Z6=XMX(K)**(-X(4))
      Z7=XMX(L)**(-X(4))
      Z4=ABS(X(3)*Z6+(1.-X(3))*Z7)
      Z=Z4**Z3
      Z1=XMX(I)-Z5*Z
      Z1Z1=Z1*Z1
      ZZ1=Z*Z1
      V=V+Z1Z1
      DX(1)=DX(1)-2.*Z2*ZZ1
      DX(2)=DX(2)-2.*XMX(T)*Z5*ZZ1
      DX(3)=DX(3)-2.*Z5*Z3*(Z4**(Z3-1.))*(Z6-Z7)*Z1
      DX(4)=DX(4)-2.*Z5*ZZ1*(ALOG(ABS(Z4))*X(5)/(X(4)*X(4))-Z3*(X(3)*Z6*
     1ALOG(ABS(XMX(K)))+(1.-X(3))*Z7*ALOG(ABS(XMX(L))))/Z4)
  101 DX(5)=DX(5)+2.*Z5*ZZ1*(ALOG(ABS(Z4))/X(4))
      RETURN
      END
END OF SEGMENT, LENGTH   399, NAME  EVALCES


C
      SUBROUTINE EVALCESM(X,DX,V)
C     SCHATTING CES-FUNCTIE MET MULTIPLICATIEVE STORINGSTERMEN
C
      DIMENSION X(5),DX(5),XMX(500)
C
```

```
      COMMON /EQ/XMX,NT,NN
C
C
      V=0.
      DO 100 I=1,5
  100 DX(I)=0.
      Z3=-X(5)/X(4)
      DO 101 I=1,NT
      K=NT+I
      L=2*NT+I
      IT=3*NT+I
      Z6=XMX(K)**(-X(4))
      Z7=XMX(L)**(-X(4))
      Z4=X(3)*Z6+(1.-X(3))*Z7
      Z=ALOG(XMX(I))-ALOG(ABS(X))-X(2)*XMX(IT)-Z3*ALOG(ABS(Z4))
      ZZ=Z*Z
      V=V+ZZ
      DX(1)=DX(1)-2.*(Z/X(1))
      DX(2)=DX(2)-2.*Z*XMX(IT)
      DX(3)=DX(3)-2.*Z*Z3*((Z6-Z7)/Z4)
      DX(4)=DX(4)-2.*Z*X(5)*(X(4)*((X(3)*Z6*ALOG(XMX(K))+(1.-X(3))*Z7*AL
     10G(XMX(L)))/Z4)+ALOG(ABS(Z4)))/(X(4)*X(4))
  101 DX(5)=DX(5)+2.*Z*(ALOG(ABS(Z4))/X(4))
      RETURN
      END

END OF SEGMENT, LENGTH   391, NAME  EVALCESM
```

```
C
      SUBROUTINE INVERT (A,IA,N,B,C,Z.DET)
C
C     SUBPROGRAMMA MATRIX INVERSIE VOLGENS GAUSS-JORDAN , C.A.C.M 120
C
      INTEGER P,Z
      DIMENSION A(IA,N),B(N),C(N),Z(N)
C
      EPS=DET
      DET=1.0
      DO 1 J=1,N
      Z(J)=J
    1 CONTINUE
C
      DO 7 I=1,N
      K=I
      Y=A(I,I)
      IF(I.EQ.N) GO TO 3
      P=I+1
      DO 2 J=P,N
      W=A(I,J)
      IF(ABS(W).LE.ABS(Y)) GO TO 2
      K=J
      Y=W
    2 CONTINUE
    3 DET=DET*Y
      IF(ABS(DET).LT.EPS) GO TO 12
      Y=1.0/Y
```

```
      DO 4 J=1,N
      C(J)=A(J,K)
      A(J,K)=A(J,I)
      A(J,I)=-Y*C(J)
      B(J),A(I,J)=Y*A(I,J)
    4 CONTINUE
      A(I,I)=Y
      J     =Z(I)
      Z(I)  =Z(K)
      Z(K)  =J
      DO 6 K=1,N
      IF(K.EQ.I) GO TO 6
      DO 5 J=1,N
      IF(J.EQ.I) GO TO 5
      A(K,J)=A(K,J)-B(J)*C(K)
    5 CONTINUE
    6 CONTINUE
    7 CONTINUE
C
      DO 10 I=1,N
    8 K=Z(I)
      IF(K.EQ.I) GO TO 10
      DO 9 J=1,N
      W=A(I,J)
      A(I,J)=A(K,J)
      A(K,J)=W
```

```
    9 CONTINUE
      P   =Z(I)
      Z(I)=Z(K)
      Z(K)=P
      DET=-DET
      GO TO 8
   10 CONTINUE
C
   11 RETURN
   12 N=-N
      GO TO 11
      END
END OF SEGMENT, LENGTH   403, NAME  INVERT


C
      SUBROUTINE EVAL (X,DX,V)
C
      DIMENSION XMX(500)
C
      COMMON /EV/INDEX,N,IPR/WE/IWARN
      COMMON /EQ/XMX,NT,NN
C
      DIMENSION X(N),DX(N)
C
  123 FORMAT (17H WEIBULL FUNCTION)
  124 FORMAT (21H SUM TWO EXPONENTIALS)
  125 FORMAT (26H KLEIN-PRESTON CONSUMPTION)
```

```
  126 FORMAT (33H C.E.S.-FUNCTION ADDITIVE ERRORS)
  127 FORMAT(39H C.E.S.-FUNCTION  MULTIPLICATIVE ERRORS)
  128 FORMAT (18H ZANGWILL FUNCTION)
  129 FORMAT (20H ROSENBROCK FUNCTION)
  130 FORMAT (14H WOOD FUNCTION)
C
      IF (IPR.NE.1) GO TO 500
      IPR=0
      GO TO (202,203,204,205,206,207,208,209),INDEX
  202 WRITE (2,123)
      GO TO 500
  203 WRITE (2,124)
      GO TO 500
  204 WRITE (2,125)
      GO TO 500
  205 WRITE (2,126)
      GO TO 500
  206 WRITE (2,127)
      GO TO 500
  207 WRITE (2,128)
      GO TO 500
  208 WRITE (2,129)
      GO TO 500
  209 WRITE (2,130)
C
  500 GO TO (1,2,3,4,5,6,7,8),INDEX
```

```
      1 CALL EVALWE (X,DX,V)
        GO TO 100
      2 CALL EVALEX (X,DX,V)
        GO TO 100
      3 CALL EVALKP (X,DX,V)
        GO TO 100
      4 CALL EVALCES (X,DX,V)
        GO TO 100
      5 CALL EVALCESM (X,DX,V)
        GO TO 100
      6 CALL EVALZANG (X,DX,V)
        GO TO 100
      7 CALL EVALROBRO(X,DX,V)
        GO TO 100
      8 CALL EVALWOOD (X,DX,V)
    100 RETURN
        END

END OF SEGMENT, LENGTH   120, NAME  EVAL


C
      SUBROUTINE EVALROBRO (X,DX,V)
C
      DIMENSION X(2),DX(2)
C
      P= X(2)-X(1)*X(1)
```

```
      V=100.*P*P+(1.-X(1))**2
      DX(1)=-400.*X(1)*P-2.*(1.-X(1))
      DX(2)=200.*P
      RETURN
      END

END OF SEGMENT, LENGTH    102, NAME  EVALROBRO


      SUBROUTINE EVALZANG (X,DX,V)
C
      DIMENSION X(3),DX(3)
C
      S1=X(1)+X(2)-X(3)
      S2=X(1)-X(2)+X(3)
      S3=-X(1)+X(2)+X(3)
       V=S1*S1+S2*S2+S3*S3
      DX(1)= 2.*( S1+S2-S3)
      DX(2)= 2.*( S1-S2+S3)
      DX(3)= 2.*(-S1+S2+S3)
      RETURN
      END

END OF SEGMENT, LENGTH    146, NAME  EVALZANG
```

```
C
      SUBROUTINE EVALWOOD (X,DX,V)
C
      DIMENSION X(4),DX(4)
C
      P1=X(2)-X(1)*X(1)
      P2=X(4)-X(3)*X(3)
      P3=1.-X(1)
      P4=1.-X(3)
      P5=X(2)-1.
      P6=X(4)-1.
      V=100.*P1*P1+P3*P3+90.*P2*P2+P4*P4+10.1*(P5*P5+P6*P6)+19.8*P5*P6
      DX(1)=200.*P1*(-2.*X(1))-2.*P3
      DX(2)=200.*P1+20.2*P5+19.8*P6
      DX(3)=180.*P2*(-2.*X(3))-2.*P4
      DX(4)=180.*P2+20.2*P6+19.8*P5
      RETURN
      END

END OF SEGMENT, LENGTH    221, NAME  EVALWOOD
```

References

[1] Barnes, J.G.P. "An algorithm for solving non-linear equations based on the second method". The Computer Journal, Vol 8, April 1965, nr. 1, pp. 66-72.

[2] Box, M.J. "A comparison of several current optimization methods, and the use of transformations in constrained problems". Computer Journal, Vol 9, pp. 67-77.

[2] Broyden, C.G. "A class of methods for solving non-linear simultaneous equations". Mathematics of Computation, Vol 19, 1965, pp. 577-593.

[4] Broyden, C.G. "Quasi-Newton methods and their application to function minimization". Mathematics of Computation, Vol 21, July 1967, nr. 99, pp. 368-381.

[5] Crockett, J.B. and H. Chernoff "Gradient methods of maximization". Pacific Journal of Mathematics, Vol 5, 1955, pp. 33-50.

[6] Davidon, W.C. Variable metric method for minimization . A.E.C. Research and Development Report, ANL-5990, November 1959, 21 pages.

[7] Fletcher, R. and M.J.D. Powell " A rapidly convergent descent method for minimization". The Computer Journal, Vol 6, July 1963, pp. 163-168.

[8] Fletcher, R. "A new approach to variable metric algorithms". The Computer Journal, Vol 13, 1970, pp.317-322.

[9] Goldfarb, D. A family of variable metric methods derived by variational means . Working paper (to be published in Math. of Comp.)

[10] Goldstein A.A. and J.F. Price "An effective algorithm for minimization". Numerische Mathematik, Vol. 10, 1967, pp. 184-189.

[11] Hartley, H.O. "The modified Gauss-Newton method for the fitting of non-linear regression functions by least squares". Technometrics, Vol 3, May 1961, nr. 2,pp. 269-280.

[12] Hestenes, M.R. "Multiplier and gradient methods" in Computing methods in optimization problems . L.A. Zadch, Academic Press, 1969, New York, pp. 143-163.

[13] Kowalik J. and M.R. Osborne Methods for unconstrained optimization problems . Elsevier Publishing Co. Inc., 1968, New York.

[14] Leon A. A comparison among eight known optimization procedures . Internal Working Paper, no.20, Space Sciences Laboratory, Universtity of California, Berkely, August, 1964.

[15] Pearson, J.D. On variable metric methods of minimization . Research Analysis Corp.,Technical Paper, RAC-TP-302, February, 1968.

[16] Powell, M.J.D. On the convergence of the variable metric algorithm . Report no. T.P. 382, A.E.R.E., Harwell, October 1969.

[17] Powell, M.J.D. Recent advances in unconstrained optimization . Report, no. T.P. 430, A.E.R.E., Harwell, November 1970.

[18] Rosen, E.M. A review of Quasi-Newton methods in non-linear equation solving and unconstrained optimization . National Conference of the ACM. Proceedings of the 21 st. Conference. Washington D.C., Thompson Book Co., 1966, pp. 37-41.

[19] Shanno, D.F. Conditioning of Quasi-Newton methods for function minimization . Center for Mathematical Studies in Business and Economics, University of Chicago, Report 6910 (Revised), August 1969, pp. 20. (Submitted to Math. of Comp.)

[20] Shanno D.F. and P.C. Kettler Optimal conditioning of Quasi-Newton methods . Center for Mathematical Studies in Business and Economics, University of Chicago, Report 6937, August 1969, p. 16.

[21] Weeg G.P. and G.B. Reed Introduction to Numerical Analysis . Waltham, Blaisdell Publishing Company, 1966, p. 48.

[22] Zangwill W.I. "Minimizing a function without calculating derivatives". Computer Journal, Vol 10, 1967, pp. 293-296.

[23] Ahlberg J.H., E.H. Nilson and J.L. Walsh The theory of splines and their applications . Academic Press, New York and London.

PREVIOUS NUMBERS:

EIT 1	J. Kriens *) .	Het verdelen van steekproeven over subpopulaties bij accountantscontroles.
EIT 2	J. P. C. Kleynen *)	Een toepassing van „importance sampling".
EIT 3	S. R. Chowdhury and W. Vandaele *)	A bayesian analysis of heteroscedasticity in regression models.
EIT 4	Prof. drs. J. Kriens *)	De besliskunde en haar toepassingen.
EIT 5	Prof. dr. C. F. Scheffer *) . . .	Winstkapitalisatie versus dividendkapitalisatie bij het waarderen van aandelen.
EIT 6	S. R. Chowdhury *)	A bayesian approach in multiple regression analysis with inequality constraints.
EIT 7	P. A. Verheyen *)	Investeren en onzekerheid.
EIT 8	R. M. J. Heuts en Walter H. Vandaele	Problemen rond niet-lineaire regressie.
EIT 9	S. R. Chowdhury *)	Bayesian analysis in linear regression with different priors.
EIT 10	A. J. van Reeken	The effect of truncation in statistical computation.
EIT 11	W. H. Vandaele and S. R. Chowdhury *)	A revised method of scoring.
EIT 12	J. de Blok	Reclame-uitgaven in Nederland.
EIT 13	Walter H. Vandaele	Mødscø, a computer programm for the revised method of scoring.
EIT 14	J. Plasmans *)	Alternative production models. (Some empirical relevance for postwar Belgian Economy)
EIT 15	D. Neeleman	Multiple regression and serially correlated errors.
EIT 16	H. N. Weddepohl	Vector representation of majority voting.
EIT 17	Walter H. Vandaele	Zellner's seemingly unrelated regression equation estimators: a survey.
EIT 18	J. Plasmans *)	The general linear seemingly unrelated regression problem. I. Models and Inference.
EIT 19	J. Plasmans and R. Van Straelen .	The general linear seemingly unrelated regression problem. II. Feasible statistical estimation and an application.

EIT 20	Pieter H. M. R[illegible]	goods only.
EIT 21	D. Neeleman *)	An alternative derivation of the k-class estimators.
EIT 22	R. M. J. Heuts	Parameter estimation in the exponential distribution, confidence intervals and a monte carlo study for some goodness of fit tests.
EIT 23	D. Neeleman	The classical multivariate regression model with singular covariance matrix.
EIT 24	R. Stobberingh	The derivation of the optimal Karhunen-Loève coordinate functions.
EIT 25	Th. van de Klundert	Produktie, kapitaal en interest
EIT 26	Th. van de Klundert	Labour values and international trade; a reformulation of the theory of A. Emmanuel.
EIT 27	R. M. J. Heuts	Schattingen van parameters in de gamma-verdeling en een onderzoek naar de kwaliteit van een drietal schattingsmethoden met behulp van Monte Carlo-methoden.
EIT 28	A. van Schaik	A note on the reproduction of fixed capital in two-good techniques.
EIT 29	H. N. Weddepohl	Vector representation of majority voting; a revised paper.
EIT 30	H. N. Weddepohl	Duality and Equilibrium

*) not available

EIT 1971